38537comp.DOC 1/8/00

APPARATUS AND METHODS FOR THE INSPECTION OF OBJECTS

FIELD OF THE INVENTION

The present invention relates to apparatus and methods for image processing.

BACKGROUND OF THE INVENTION

Apparatus and methods for analyzing images, and in particular for image processing and analysis, which are useful in inspection of patterned objects, are well known in the art.

The following references describe image processing methods which may be useful in understanding the present invention:

C. Gonzalez and P. Wintz, Digital Image Processing, Addison Wesley, Reading, MA, 1987; and

John C. Russ, The Image Processing Handbook, CRC Press, 1994.

The following references describe edge detection methods which may be useful in understanding the present invention:

D. Marr and E. Hildreth, Theory of Edge Detection, Proceedings of the Royal Society of London; and

M. Chapron, "A new chromatic edge-detector used for color image segmentation", 11th APR International Conference on Pattern Recognition.

The following references describe color image segmentation methods which may be useful in understanding the present invention:

Philippe Pujas and Marie-Jose Aldon, "Robust Colour Image Segmentation", 7th International Conference on Advanced Robotics, San Filiu de Guixols, Spain, September 22, 1995, and

Leila Shararenko, Maria Petrou, and Josef Kittler, "Automatic Watershed Segmentation of Randomly Textured Colour Images, IEEE.

US Patent 4,758,888 to Lapidot describes a method of and means for inspecting workpieces traveling along a production line, including on-line inspection for flaws without interrupting the progress or workpieces along the production line.

US Patent 5,586,058 and US Patent 5,619,429 to Aloni *et al.* describe apparatus and methods for inspecting patterned objects and detecting defects therein, including inspecting a binary level representation of the object, inspecting a gray level representation of the object, and preferably reinspecting the grayscale representation of the object to filter false alarms and to classify defects.

US Patent 5,774,572 to Caspi describes an automatic visual inspection system, which is operative to convolve a 2-dimensional digital gray scale image of an object with a filter function related to the second derivative of a Gaussian function forming a 2-dimensional convoluted image have signed values. The location of an edge in the object is achieved by finding zero crossings between adjacent oppositely signed values.

US Patent 5,774,573 to Caspi *et al.*. describes a visual inspection system which uses convolution of a 2-dimensional digital gray scale image of an object with a filter function related to the second derivative of a Gaussian function forming a 2-dimensional convolved imaged having signed values. The convolution of Caspi *et al.*can be performed with a difference-of-two-Gaussians, one positive and one negative.

PCT Application IL98/00393 describes inspection of printed circuit boards using color, including the use of color to identify certain types of conditions, such as conductor oxidation, using color.

PCT Application IL98/00477 describes methods for analytically representing an image so that morphological operations such as dilation, erosion, and scale measurement may be performed on non-binary pixels, preferably more efficiently than using previous methods.

Applicant's unpublished Israel Patent Application No. 125929 describes article inspection systems and methods, including an improved image acquisition system. Methods which may be useful in image analysis are described in the following publication:

Dorin Comaniciu and Peter Meer, "Distribution Free Decomposition of Multivariate Date", SPR '98 Invited Submission, Department of Electrical and Computer Engineering, Rutgers University, Piscataway, NJ 08855, USA.

The disclosures of all publications mentioned above and throughout the specification and of the publications cited therein are hereby incorporated by reference.

SUMMARY OF THE INVENTION

The present invention seeks to provide improved apparatus and methods for image processing.

The present invention is useful in conjunction with the optical inspection system described in Applicant's Israel Patent Application IL 131092 entitled "Optical Inspection System", filed 25 July 1999, the disclosure of which is incorporated herein by reference.

Inputs to the image processing system shown and described herein, typically comprise at least one of the following:

a. Optionally, snapshots of areas or regions of interest, including some or more of the following: areas defined in advance by a user, areas surrounding suspected defects, and areas surrounding predetermined types of features;

b. Binary CEL data (sometimes termed herein "CEL data"), which is data that identifies contour elements which correspond to borders between different regions in a pattern;

c. Color CEL data, namely data identifying contour elements which correspond to borders between different regions in a pattern and identifying a type of region on either side of the CEL;

d. Morphological feature inspection triggers;

e. Color defect inspection triggers;

f. Binary defect inspection triggers (sometimes termed herein "hardware defects"), which are nicks, protrusions, and CELs on skeletons, wherein CELs are located on morphological skeletons; and

f. User identified area of interest inspection triggers;

Preferably, theses inputs are provided by an image processing unit substantially as described in Applicant's Israel Patent Application IL 131092.

There is thus provided in accordance with a preferred embodiment of the present invention a method for inspecting objects, the method including creating a reference image for a representative object, the reference image including an at least partially vectorized first representation of boundaries within the image, acquiring an image of an object under inspection including a second representation of boundaries within the image, and comparing the second representation of boundaries to the at least partially vectorized first representation of boundaries, thereby to identify defects.

Further in accordance with a preferred embodiment of the present invention the comparing step employs a user-selected variable threshold for acceptable distance between corresponding portions of the boundaries in the first and second representations.

There is further provided in accordance with another preferred embodiment of the present invention, a system for image processing including a boundary identifier operative to generate a representation of boundaries of known elements in the image, a hardware candidate defect identifier operative to identify candidate defects in the image, in hardware, a software candidate defect inspector receiving an output from the hardware candidate defect identifier and using the representation of boundaries to identify at least one false alarm within the output, in software. Preferably the identification of the at least one false alarm is effected by comparing the boundaries to a reference.

Still further in accordance with a preferred embodiment of the present invention the boundary identifier includes a hardware boundary identifier operative to generate a representation of boundaries of known elements in the image, in hardware.

Additionally in accordance with a preferred embodiment of the present invention the system also includes a software candidate defect identifier operative to identify additional candidate defects in the image, in software.

Also in accordance with a preferred embodiment of the present invention the software candidate defect inspector also receives a second output from the software

candidate defect identifier and uses the representation of boundaries to identify at least one false alarm within the second output, in software.

Further in accordance with a preferred embodiment of the present invention the hardware candidate defect identifier employs the representation of boundaries in order to identify at least some candidate defects.

Still further in accordance with a preferred embodiment of the present invention the software candidate defect identifier employs the representation of boundaries in order to identify at least some candidate defects.

There is additionally provided in accordance with another preferred embodiment of the present invention a system for image processing including a learning subsystem operative to define first and second areas in an object under inspection wherein the first areas each include at least one known critical object element and the second areas include no such known critical object elements, and a defect detector operative to inspect the first areas using a first procedure based on prior knowledge regarding the known critical object elements and to inspect the second areas using a second procedure which differs from the first procedure.

Further in accordance with a preferred embodiment of the present invention the second procedure includes a hardware inspection of a second area operative to identify candidate defects in the second area and a subsequent software inspection of the second area, only if at least one candidate defects are found in the second area, and operative to analyze the at least one candidate defects found in the second area and to identify false alarms therewithin.

There is also provided in accordance with another preferred embodiment of the present invention a method for inspecting an object including, in a first stage of inspection, identifying a location of at least one candidate defect, and for each candidate defect, determining of a candidate area, at the location, for inspection at least of the size and shape of the contour area being based, at least in part, on an output of the identifying step, and in a second stage of inspection, inspecting each candidate area to confirm the defect.

Additionally in accordance with a preferred embodiment of the present invention the first stage is carried out in hardware.

Further in accordance with a preferred embodiment of the present invention the second stage is carried out in software.

Still further in accordance with a preferred embodiment of the present invention the step of determining includes determining the size of a candidate area at a known location.

Additionally in accordance with a preferred embodiment of the present invention the second stage includes performing different inspections depending on criteria of the candidate area or of the candidate defect.

Also in accordance with a preferred embodiment of the present invention the second stage includes performing different inspections depending on characteristics of the at least one candidate defect as identified in the identifying step.

Further in accordance with a preferred embodiment of the present invention the second stage includes performing different inspections depending on functionality of the object portion in which the candidate defect resides.

Still further in accordance with a preferred embodiment of the present invention the second stage includes performing different inspections depending on the degree of criticality of the functionality of the object portion in which the candidate defect resides.

There is additionally provided in accordance with another preferred embodiment of the present invention a modular image processing system with user-customizable image analysis functions, for use in conjunction with a scanner, the system including, an image processing engine receiving at least one stream of image data to be analyzed from the scanner, and an engine configurator operative to receive a sequence of definitions of at least one user-customized image analysis function to be performed on the image data by the image processing engine, wherein the image processing engine is operative to analyze at least one channel of image data in accordance with each definition in the sequence of definitions fed into the engine configurator, including performing different image analysis functions, differing from one another more than only as to parameters, depending on a current definition arriving at the engine configurator from among the sequence of definitions.

There is also provided in accordance with another preferred embodiment of the present invention a method for automatically optically inspecting an object,

including in a first inspection step, defining a plurality of regions of interest for image processing, the regions of interest including at least one region of interest defined by a user and at least one region of interest automatically defined by optically inspecting the article, and providing to an image information correlating to an area surrounding each region of interest, and in a second inspection step, automatically processing information for the region of interest to determine the presence of defects in the article.

Further in accordance with a preferred embodiment of the present invention the image of an area surrounding a region of interest is smaller than an image of the object.

Still further in accordance with a preferred embodiment of the present invention the region of interest automatically defined by optically inspecting the article includes a candidate defect in a pattern formed on the object.

Additionally in accordance with a preferred embodiment of the present invention the region of interest automatically defined by optically inspecting the article includes a predetermined morphological feature formed in a pattern on the object.

Also in accordance with a preferred embodiment of the present invention, the providing step includes providing to the image processor parsed information relating to boundaries in the region of interest. Preferably, the providing step additionally includes provding a color image of the region of interest to the image processor.

Further in accordance with a preferred embodiment of the present invention the automatically processing step includes applying, to regions of interest defined by a user, at least one image processing method that is different from an image processing method applied to regions of interest automatically defined by optically inspecting the article.

Still further in accordance with a preferred embodiment of the present invention the providing step includes identifying the type of defect in the region of interest, and the automatically processing step includes applying an image processing method that is suited for the type of defect in the region of interest.

Additionally in accordance with a preferred embodiment of the present invention the providing step includes identifying the type of morphological feature in the region of interest, and the automatically processing step includes applying an image

processing method that is suited for the type of morphological feature in the region of interest.

Also in accordance with a preferred embodiment of the present invention the at least one region of interest defined by a user is defined prior to optically inspecting the object in a software definition step, the at least one region of interest automatically defined by optically inspecting the article is performed in a hardware inspection step, and the automatically processing each image of a region of interest is performed in a software image processing step.

There is further provided in accordance with another preferred embodiment of the present invention, a method for inspecting ball grid array substrates, the method including generating at least one model for at least one feature in a reference image of a ball grid array substrate, and storing the model in memory, acquiring an image of the ball grid array substrate, and inspecting predetermined regions of the image of the ball grid array substrate to determine whether a feature in the predetermined region fits the model.

Further in accordance with a preferred embodiment of the present invention the feature comprises a circle.

Still further in accordance with a preferred embodiment of the present invention the model of the circle includes a center point at a predetermined location and a radius within a predefined tolerance.

Additionally in accordance with a preferred embodiment of the present invention the

parameters of the model are at least partly adjustable in an off-line mode prior to inspection.

Also in accordance with a preferred embodiment of the present invention the feature includes a bonding pad.

BRIEF DESCRIPTION OF THE DRAWINGS AND APPENDICES

The present invention will be understood and appreciated from the following detailed description, taken in conjunction with the drawings and appendices in which:

Fig. 1A is a simplified block diagram of an image processing system constructed and operative in accordance with one preferred embodiment of the present invention;

Fig. 1B is a simplified functional block diagram of an image processing system constructed and operative in accordance with another preferred embodiment of the present invention;

Fig. 2 is a simplified functional block diagram of the apparatus of Fig. 1B when the user-selected scenario is "design rule" mode;

Fig. 3 is a simplified functional block diagram of a preferred implementation of unit 210 of Fig. 2;

Fig. 4 is a simplified functional block diagram of the apparatus of Fig. 1B when the user-selected scenario is "learn" mode;

Fig. 5 is a simplified functional block diagram of a preferred implementation of an individual one of the "learn mixed" units of Fig. 4;

Fig. 6 is a simplified functional block diagram of a preferred implementation of an individual one of the three blocks in Fig. 5;

Figs. 7A - 7B, taken together, form a table describing preferred test functions provided for each window type in the course of a learn scenario;

Fig. 8 is a table of preferred functions typically executed by the task multi slice manager of Fig. 5 in the course of a learn scenario;

Fig. 9 is an execution graph of a scan scenario termed herein "learn.ref.config";

Fig. 10 is an execution graph of a scan scenario termed herein "learn.ref.eg.config";

Fig. 11 is an execution graph of a scan scenario termed herein "inspect.mixed.config";

Fig. 12 is an execution graph of a scan scenario termed herein "inspect.mixed.egmain.config";

Fig. 13 is an execution graph of a scan scenario termed herein "inspect.mixed.egside.config";

Fig. 14 is an execution graph of a scan scenario termed herein "inspect.mixed.eg.config";

Fig. 15A - 15C, taken together, form a table describing preferred test functions provided for each window type in the course of an inspect scenario;

Fig. 16 is a table of preferred functions typically executed by the task multi slice manager of Fig. 12 in the course of an inspect scenario;

Fig. 17 is a diagram of a preferred data structure for a report element as may be received from the hardware image analysis system of copending Israel Patent Application No. 131092;

Fig. 18 is a preferred data structure for the header of Fig. 17;

Fig. 19 is an example of a simplified execution graph for the image processing operating system SIP of Fig. 1;

Fig. 20 is another example of a simplified execution graph which differs from the graph of Fig. 19 in that it is adapted for execution by a computer farm comprising at least two simultaneous parallel processors;

Fig. 21A is a conceptual illustration of a pixel including one CEL;

Figs. 21 B –21D is a conceptual illustration of the relationship between features and their analytical representation by CELs and vectorized line components.

Figs. 22A - 22H is an illustration of an edge code preferably used to define the type of edge or edges included in an individual record;

Figs. 23A - 23H is an illustration depicting cell orientations for edge code values 0 to 9 respectively, and dir = 1;

Fig. 23I is illustration of a CEL defined by the following parameters: edgecode = 0, first = 14, last = 0;

Fig. 24 is a pictorial illustration of an envelope data structure surrounding a CEL;

Fig. 25 is a pictorial illustration of two envelopes enveloping two consecutive line segments respectively;

Fig. 26 is a simplified pictorial illustration of a collection of identical frames collected on a panel for inspection in accordance with a preferred embodiment of the present invention;

Fig. 27 is a pictorial illustration of an example of data that may exist inside a window;

Figs. 28A - 28B are pictorial illustrations of preferred form of line width measurement for parallel and non-parallel lines respectively;

Fig. 29 is a pictorial illustration of a transformation between an on-line and a reference coordinate system;

Fig. 30A is a pictorial illustration of a vectorized polygonal reference correlating to edge contours;

Fig. 30B is a pictorial illustration of on-line CELs overlaid over the polygonal reference of Fig. 30A, with perfect registration between on-line and reference data;

Fig. 30C is a pictorial illustration of on-line CELs, different from on-line CELs in Fig. 30B, overlaid over the polygonal reference of Fig. 30A, with perfect registration between on-line and reference data;

Fig. 31 is a simplified flow chart showing and describing a preferred method of micro-registration in accordance with a preferred embodiment of the present invention;

Fig. 32A is a pictorial illustration of on-line CELs overlaid over polygonal reference, with non-perfect registration between on-line data and the reference data;

Fig. 32B is a pictorial illustration of the structure of Fig. 31A after performance of a registration method in accordance with a preferred embodiment of the present invention;

Fig. 33 is a flow chart showing the function of an image processing system constructed and operative in accordance with another preferred embodiment of the present invention;

Figs. 34-42 are screen displays from a general user interface used to define of parameters for inspection of an image by an image processing system

constructed and operative in accordance with a preferred embodiment of the present invention;

Fig. 43 is a simplified block diagram showing relationships between data sources and a processing module constructed and operative in accordance with a preferred embodiment of the present invention;

Fig. 44 is a simplified block diagram showing relationships between data sources and a pre-emptive processing module constructed and operative in accordance with a preferred embodiment of the present invention;

Fig. 45 is a simplified block diagram showing user defined tasks of a processing module, constructed and operative in accordance with a preferred embodiment of the present invention;

Fig. 46 is a simplified block diagram illustrating a composite processing module constructed and operative in accordance with a preferred embodiment of the present invention;

Fig. 47 is a simplified block diagram illustrating input channels into a pre-emptive processing module and a cooperative processing module, constructed and operative in accordance with a preferred embodiment of the present invention;

Fig. 48 is a simplified block diagram illustrating output channels from a pre-emptive processing module and a cooperative processing module, constructed and operative in accordance with a preferred embodiment of the present invention;

Fig. 49 is a simplified block diagram illustrating a registration processing module constructed and operative in accordance with a preferred embodiment of the present invention;

Fig. 50 is a simplified block diagram illustrating a a root process manager, constructed and operative in accordance with a preferred embodiment of the present invention;

Fig. 51 is a simplified block diagram illustrating an execution graph scenario constructed and operative in accordance with a preferred embodiment of the present invention;

Fig. 52 is a simplified block diagram illustrating a root directory tree from which the SIP begins, constructed and operative in accordance with a preferred embodiment of the present invention;

Fig. 53 is a simplified flow chart illustrating SIP states and the commands that effect them, constructed and operative in accordance with a preferred embodiment of the present invention;

Fig. 54 is a simplified illustration of a single snap covering the area around a single Color_defect report, constructed and operative in accordance with a preferred embodiment of the present invention;

Fig. 55 is a simplified illustration of snaps covering the area around multiple Color_defect reports, constructed and operative in accordance with a preferred embodiment of the present invention;

Fig. 56 is a simplified schematic illustration of the structure of a line of 48 snap reports, constructed and operative in accordance with a preferred embodiment of the present invention;

Fig. 57 is an example of part of a Ball Grid Array substrate to be inspected by an inspection system constructed and operative in accordance with a preferred embodiment of the present invention;

Fig. 58 is a close-up view of the lower end of the Ball Grid Array Substrate of Fig. 57;

Fig. 59 is a close-up view of a bond finger on the Ball Grid Array substrate of Fig. 57 as viewed by an inspection system constructed and operative in accordance with a preferred embodiment of the present invention;

Fig. 60A is a flow chart illustrating data flow through a PIM map generator during the Learn stage of an inspection scan, constructed and operative in accordance with a preferred embodiment of the present invention;

Fig. 60B is a flow chart illustrating data flow through a filter during the Inspect stage of an inspection scan, constructed and operative in accordance with a preferred embodiment of the present invention;

Fig. 61 is a simplified block diagram illustrating subtraction of regions by the map generator, constructed and operative in accordance with a preferred embodiment of the present invention;

Fig. 62 is a simplified block diagram illustrating the data structure of a PIM stored in runlength format, constructed and operative in accordance with a preferred embodiment of the present invention;

Fig. 63 is a simplified block diagram of the general structure of the PIM system, constructed and operative in accordance with a preferred embodiment of the present invention;

Fig. 64 is a simplified block diagram illustrating inputs and outputs of a task trigger handler, constructed and operative in accordance with a preferred embodiment of the present invention;

Fig. 65 is a simplified flow chart illustrating the function of a task trigger handler, constructed and operative in accordance with a preferred embodiment of the present invention;

Fig. 66 is a simplified illustration of the run length structure of a task trigger handler where the map region is the bounding rectangle enlarged by two pixels on either side, constructed and operative in accordance with a preferred embodiment of the present invention;

Fig. 67A is a simplified illustration of the snap rectangular regions cut according to the clusters defined in Fig. 66;

Fig. 67B is a simplified illustration of the snap rectangular windows possessing the cluster, and a rectangle with RGB data, after processing of the clusters defined in Fig. 66;

Fig. 68 is a simplified illustration of the general configuration of a snap area treated by the process termed herein "task snap2ppm" that enables the separation of snap data into separate files each containing only one snap, constructed and operative in accordance with a preferred embodiment of the present invention;

Fig. 69 is a simplified illustration of examples of configurations of snap areas not treated by the task snap2ppm, constructed and operative in accordance with a preferred embodiment of the present invention;

Fig. 70 is a simplified illustration of a chain of blocks structure in a windowing reference management tool, constructed and operative in accordance with a preferred embodiment of the present invention;

Fig. 71 is a simplified illustration of the indexing main structure of a windowing reference management tool, constructed and operative in accordance with a preferred embodiment of the present invention;

Fig. 72 is a simplified illustration of fast retrieval of data via a windowing query, constructed and operative in accordance with a preferred embodiment of the present invention;

Fig. 73 is a simplified illustration of an electrical circuit on a board whose image is to be analyzed;

Fig. 74 is two additional simplified illustrations of a the board of Fig. 73

Fig. 75 is a polyline object illustrating the deviation parameter of a process for straight line approximation of connected components composed of CELs, constructed and operative in accordance with a preferred embodiment of the present invention;

Fig. 76 is a polyline object illustrating the deflection parameter of a process for straight line approximation of connected components composed of CELs, constructed and operative in accordance with a preferred embodiment of the present invention;

Fig. 77 is a graph illustrating the small CEL length parameter of a process for straight line approximation of connected components composed of CELs, constructed and operative in accordance with a preferred embodiment of the present invention;

Fig. 78 is a polyline object illustrating a 'quality assurance' mechanism in a process for straight line approximation of connected components composed of CELs, constructed and operative in accordance with a preferred embodiment of the present invention;

Fig. 79 is a polyline object illustrating an advanced 'quality assurance' mechanism in a process for straight line approximation of connected components composed of CELs, constructed and operative in accordance with a preferred embodiment of the present invention;

Fig. 80 is a graph illustrating the results of a performance test in a process for straight line approximation of connected components composed of CELs, constructed and operative in accordance with a preferred embodiment of the present invention;

Fig. 81 is a graph derived from the graph of Fig. 80 by examining the situation where the compression ratio (as a function of deflection angle) becomes constant for every vectorization style; and

Fig. 82 is a graph derived from the graph of Fig. 80 by examining the deflection angle where the compression ratio (as a function of deflection angle) becomes constant for every vectorization style, and the deflection angle that produces minimum time vectorization.

Attached herewith are the following appendices which aid in the understanding and appreciation of one preferred embodiment of the invention shown and described herein:

Appendix A is a description of a preferred OOD design of symmetric multi-processing (SMP) capable SIP;

Appendix B is a description of a preferred structure of snap reports;

Appendix C is a description of a filter tolerance setting PIM constructed and operative in accordance with a preferred embodiment of the present invention;

Appendix D is a description of a SIP camera model, constructed and operative in accordance with a preferred embodiment of the present invention;

Appendix E is a description of registration and line width measurements constructed and operative in accordance with a preferred embodiment of the present invention;

Appendix F is a description of a preferred method for packing snap data;

Appendix G is a description of a preferred method for building PPM files for viewing snap data;

Appendices H and I, taken together are a description of structures and methods associated with the windowing of reference raw data, constructed and operative in accordance with a preferred embodiment of the present invention;

Appendix J is a description of a preferred method for performing the balls measurement task;

Appendix K is a description of a preferred method for creating connected components from CELs; and

Appendix L is a description of a preferred method for approximating connected components by straight lines.

DETAILED DESCRIPTION OF PREFERRED EMBODIMENTS

Overview

In the following specification, although reference is extensively made to BGAs (ball grid array substrates) for the purposes of illustrating the present invention, it is readily appreciated that the invention is applicable to the inspection of any suitable patterned object. Thus, the term BGA, as used hereinbelow, shall not be limited solely to ball grid array substrates, but rather shall be considered exemplary and deemed to refer to include any suitable form of patterned object, such as printed circuit board substrates, laminated printed circuit boards, lead frames, flat panel displays, hybrid chip packaging substrates, tape automated bonding substrates, and any other suitable patterned object including various etched and engraved metal substrates as may be used, for example, in medical implants.

In the drawings, unless indicated otherwise, rectangles denote functional units whereas rounded blocks denote data processed by the functional units.

Reference is now made to Figure 1A which is a simplified block diagram of an image processing system constructed and operative in accordance with a preferred embodiment of the present invention.

In this embodiment, scanner 2 preferably optically scans object 4 and generates an RGB (Red, Green and Blue) two-dimensional image representing the object 4, and transmits RGB image information to an image processor 5. Preferably a spatial area of interest identifier 6 is provided, and image processor 5, spatial area of interest identifier 6 and line width and registration processor 3 output additional inspection triggers and other data to a SIP unit 7 located downstream which performs additional inspection functionalities not provided by image processor 5, preferably inspection functionalities that are dynamically adapted to an inspection trigger input so that each different type of inspection trigger causes a region associated with the trigger to be inspected with an inspection routine that is best suited for inspecting the region triggered.

Scanner 2 is preferably an image illumination and acquisition system as described in Applicant's copending U.S. patent application _____ entitled

Illumination for Inspecting Surfaces of Objects", and filed on May 5, 2000. Although scanner 2 is preferably operative to acquire RGB optical images, it is readily appreciated that scanner may be operative to acquire other forms of image, for example monochrome gray level images, images in HIS (Hue, Saturation and Intensity) format, or images correlating to non-optical properties of the object such as electric fields surrounding a printed circuit boards which define certain electrical properties relating to electrical circuits on the printed circuit board. It is appreciated that the present invention may relate to image processing of any suitable optical or non-optical image.

In accordance with a preferred embodiment of the invention, image processor 5 preferably is operative to receive image information data, to parse image information data into fundamental image data and to provide triggers based on the detection of suspected defects and predetermined features for additional inspection in a downstream inspection unit. Image processor 5 preferably is a multi-module processing unit that comprises one or more of the following modules:

(i) A binary CEL identifier 8 module. Binary CELs are contour elements that represent edge boundaries between bordering regions in an object 4, for which a high gradient in the level of reflective intensity exists between darker and lighter regions in a monochrome image. Binary CELs are referred to hereinbelow either as "CELs" or "binary CELs". Binary CEL identifier 8 receives and processes image data from scanner 2 and outputs a report of binary CELs existing in an image of object 4.

(ii) A color CEL identifier 10. Color CELs represent edge contours between bordering regions or different color populations in a pattern on the surface of object 4. A color population is region of substantially homogeneous color, and is preferably correlated to a material. Color CELs are generated with respect to color populations chosen from among a number of predetermined color populations, and which identify which of the color populations exist on either side of the edge. Color CEL identifier 10 preferably processes image data received from scanner 2 to identify the different color populations in the image data, correlates the color populations to materials known to be in the object, and outputs a report of color CELs existing in an image of object 4 which define regions of different materials and the borders between the regions.

(iii) A morphological feature identifier 11. Morphological features are predetermined features which are identifiable in a pattern. Morphological feature identifier 11 preferably is operative to receive inputs from one or more of image data from scanner 2, binary CEL information from binary CEL identifier 8, and color CEL information from color CEL identifier 10. Morphological feature identifier 11 processes these inputs, and outputs a report of morphological features, which trigger further inspection in a downstream inspection unit. Preferably, at least some of the morphological features are related to materials in object 4. In the context of BGA inspection, morphological features include for example: conductor pads, open ends, shorts, junctions, islands and other similar features as may be selected by an operator.

(iv) A Defect Identifier 13. Defects are predefined types of suspected defects which are identifiable in a pattern. Defect identifier 13 is operative to receive inputs from one or more of image data from scanner 2, binary CEL identifier 8 and color CEL identifier 10. Defect identifier 13 processes these inputs and outputs a report of suspected defects which trigger additional inspection in a downstream inspection unit. Preferably, suspected defects that are reported by defect identifier 13 are defects that are identified with reference to a set of predetermined design rules, and are identified without comparing the input data to a reference. In the context of BGA inspection, suspected defects comprise, for example: unevenness in contours defining conductors which exceed a predefined parameter of evenness, oxidation, low contrast color surface defects such as chemical residues, and high contrast surface defects such as scratches.

(v) An optional snap generator 14. When present, snap generator 14 preferably is operative, in response to feature and defect triggers received from morphological feature identifier 11 and defect identifier 13, to generate snapshot images of selected areas in the image. Snap images are portions of images received from the scanner 2 which preferably are generated as a 'window of interest' surrounding each inspection trigger received from morphological feature identifier 11 and defect identifier 13. Additionally, snap generator 14 may generate snap images for spatial regions of interest in accordance with triggers received from a spatial area of interest identifier 6 as described in greater detail hereinbelow.

Preferably, the shape and dimension of snap images output by snap generator 14 are dynamically structured in response to a trigger according to the specific type of region of interest. Snap generator 14 preferably is operative to output image data which is readily converted into a dynamic image in a downstream process. A preferred hardware implementation of an image processor suitable for use as image processor 5 is described in Applicant's copending patent application IL 131092. It is readily appreciated that the above description is organized around the preferred image processing functionalities existing in image processor 5. However, the function of image processor 5 may in fact be grouped into modules, for example binary processing, color processing and morphological processing, which issue functional reports grouped as described above.

Reports may be grouped into various reporting channels depending on hardware architecture. Accordingly, in a preferred embodiment of the invention, image processor 5 generates reports which are provided to the SIP 7 in two channels, wherein each report channel comprises a number of reports. Preferably, a first channel, designated the "Snap Channel" comprises a report of defect data relating to color surface defects from the defect identifier 13 and a report which comprises snap images of regions of interest from snap generator 14. A second channel, designated the "CEL" channel, preferably comprises a report of binary CELs from binary CEL identifier 8, and report called a defect report which reports color CELs from color CEL identifier 10, morphological features from morphological feature identifier 11, and binary defects, such as nicks and protrusions, also from defect identifier 13.

In accordance with a preferred embodiment of the present invention, a spatial area of interest identifier 6 is provided to define recurring spatial areas of interest in an inspected object. Spatial areas of interest are regions existing in object 4, in which a specific type of feature is expected to be present, and for which a predefined inspection routine tailored to the specifc type of feature is typically performed.

Preferably, the spatial area of interest generator 6 is operative, in an off-line "learn" mode, described in greater detail below, prior to defect inspection. In a preferred embodiment, spatial area of interest identifier 6 comprises two modules: a user defined areas of interest module 28 and a computer generated area of interest module 9.

Preferably, in the learn mode, via a general user interface described in greater detail with respect to Figs. 34–42 hereinbelow, which is part of user defined area of interest module 28, a user defines general regions of inspection interest which are located on objects to be inspected 4 in a batch of objects. Preferably, the user also identifies a type of structure which is expected to be located in area of interest. Accordingly, in the context of inspection of BGAs, the user may define a general region that comprises bonding pads, targets, chip areas or balls.

In computer generated areas of interest module 9, a computer processor, which may in an alternative embodiment comprise a part of the image analyzer 24 in the SIP 7, "learns" the location of the structures identified by the user as structures of interest in the spatial region defined by the user in the user defined area of interest module 28. Computer generated areas of interest module 9 preferably generates a report which specifies a localized spatial area around closely surrounding each of the structures in the area defined by the user. Accordingly, in the context of BGA inspection, if the user defines a general region including bonding pads, the computer generated area of interest module "learns" the structure and location of each bonding pad in the region defined by the user, and wherever a bonding pad is expected to be located, the computer generated area of interest module defines a localized region of interest. The localized spatial areas of interest defined around each structure in the user defined area are saved in a memory, preferably associated with the SIP 7 (not shown) for later access during inspection .

In accordance with a preferred embodiment of the present invention, SIP 7 is provided downstream of image processor 5, and is operative to receive image data from image processor 5 and to perform inspection routines that are not provided by image processor 5. For example, SIP 7 is operative in part to provide a comparison mode inspection in which parsed image data from an object being inspected 4 is compared to reference data for the object. Additionally, SIP is operative in part to dynamically adapt inspection routines according to predefined rules correlating to spatial regions according to stored information from the spatial area of interest identifier 6, to types of features that are detected during image processing by image processor 5, and to types of defects that are detected during image processing by image processor 5.

Preferably SIP 7 is a software based image processor unit running customized software image processing routines on a dedicated workstation, e.g. a microcomputer workstation operating in a LINUX™ environment, and comprises the following modules: (i) a line width and registration processor module 3, (ii) a task packer and window of interest generator module 12, and (iii) an analyzer module 15.

In accordance with a preferred embodiment of the invention, line width and registration processor 3 receives binary CEL information from binary CEL identifier 8 and morphological feature data from morphological feature identifier 11, and processes the input information in the following modules: (i) a registration generator module 18 operative to perform registration transformations in a transform calculator 20 to register an image of an object under inspection 4 to a reference image, and to determine the presence and absence of features in an excess/missing calculator 22 to determine whether all the expected features, and only those expected features, are present in an object under inspection 4; and (ii) a line width calculator 16 operative to receive registration transform data from transform calculator 20 to calculate line widths for conductors and to verify that the line width at each particular location is within a permitted range for the location, and to verify that the space between conductors is not smaller than a predetermined global value.

The configuration and operation of a preferred registration generator 18 and its respective transform calculator 20 and excess/missing calculator modules 22 are described in greater detail below. The configuration and operation of a preferred line widths calculator 16 is described in greater detail with reference to Appendix E

In accordance with a preferred embodiment of the present invention, the task packer/window of interest generator unit 12 of SIP 7 preferably receives inputs from each of the following modules: binary CEL identifier 8, color CEL identifier 10, morphological feature identifier 11, defect identifier 13, transform calculator 20, excess missing calculator 22, and line width calculator 16. In addition task packer 12 receives spatial areas of interest that were defined in a learn mode by spatial area of interest identifier 6, and subsequently stored in memory.

Task packer 12 is operative to generate a window of interest surrounding various inspection triggers. Inspection triggers preferably are predetermined types of features received from morphological feature identifier 11, predetermined types of

defects received from defect identifier 13, and predetermined spatial areas of interest identified by spatial area of interest identifier 6, and predetermined line width defects, excess/missing defects received from line width and registration processor 3, and stored in memory. Optionally, each inspection trigger is provided with a snap image from snap generator 14 of a region surrounding the trigger. Preferably, the shape and dimensions of the snap image are produced dynamically in response to the type and/or the location of a trigger in accordance with a predetermined rule. The shape and dimensions of the snap image may also be standard for generally all inspection triggers.

For each inspection trigger, task packer 12 preferably generates a package which comprises the identification of an appropriate inspection task which is chosen dynamically in order to perform a particular inspection task in response to a particular type of inspection trigger that is received, and the appropriate image information data necessary to perform the task chosen. For example, in the context of BGA inspection, a package for a bonding area defined as a spatial area of interest preferably comprises identification of an appropriate bonding area inspection task along with a complete set of binary CEL, color CEL and morphological feature date for the spatial region. A package for regions identified as having a color surface defect preferably comprises an identification of a color surface defect inspection task and a snap image of the affected area, but no CEL information. A package for a region identified as having a nick/protrusion defect preferably comprises an identification of an appropriate nick/protrusion inspection method, binary CELs for the affected region and optionally a snap image.

Task packer 12 typically transmits a dynamic processing package, comprising image and method information, represented by a large arrow, as seen in Fig. 1B, to the analyzer unit 15. Dynamic information is preferably only transmitted to analyzer 15 for specific, defined regions of the image, based upon defect, feature and area of interest information received by the task packer 12. This "pre-processing" significantly reduces the amount of image data to be analyzed by analyzer unit 15, and hence lowers the computational load, which ultimately reduces inspection time costs associated with analyzing an object.

In accordance with a preferred embodiment of the present invention analyzer unit 15 preferably comprises two units: an image analyzer module 24, and a post processor module 26.

Image analyzer module 24 preferably is operative to receive the dynamic processing package and analyze the information contained therein by employing the method identified by task packer 12. Typically, the inspection methods identified by task packer 12 and performed by image analyzer 24 are comparison based inspection tasks in which a region in the image surrounding each inspection trigger is analyzed and compared to a reference. Comparison is preferably based on a comparison of CELs in the image of the object being inspected 4 to CELs in a reference image for the region correlating to the region triggered as having a suspected defect.

The post processor 26 preferably is operative to implement the defect filter of Figure 33, described below. It is also preferably operative to analyze snapshot images from scratch with respect to regions that are determined by image analyzer 24 to comprise defects of a predetermined type for which there is a high probability of the suspected defect being a false positive.

SIP 7 functions substantially as described herein, and preferably provides an output report 30 of defects and features of the object based upon the windows of interest provided by the window of interest generator.

Reference is now made to Figure 1B which is a simplified functional block diagram view of a BGA (ball grid array) substrate inspection system constructed and operative in accordance with another preferred embodiment of the present invention. The BGA inspection system of Fig. 1B is similar to the preferred embodiment described with respect to Fig. 1A, however it is notably different in that the BGA inspection system seen in Fig. 1B shows inputs from three sensors, or cameras, which are part of scanner 34 (corresponding to scanner 2 in Fig. 1A), and does not include a snap generator 14. It is appreciated that reference to an inspection system for inspection of BGAs is intended for illustration purposes, and that any suitable object may be inspected.

As seen, the inspection system of Fig. 1B comprises a user workstation 32, a scanner 34 which typically generates an RGB image or other digital image of a given panel to be inspected 36, and an HIP (Hardware Image Processing) unit 38,

typically comprising dedicated hardware that processes images generated by the scanner 34. It is appreciated that while HIP is preferably implemented in dedicated hardware, HIP may also be implemented in a DSP environment, or in custom software running on a general purpose computer workstation. As discussed HIP unit 38 (which corresponds to image processor 5 in Fig. 1A) typically generates the following outputs, as described in detail below:

a. Binary CELS

b. Morphological features

c. Color CELS;

d. Color defects;

e. Snapshots of RGB images around selected defects.

Downstream of the HIP unit 38 is a SIP (Software Image Processing) unit 40 which receives the above 5 inputs from the HIP 38 unit and performs image processing operations as described in detail below. It is appreciated that while SIP is preferably implemented in software, it may be implemented in other at least partially programable environments, such as DSPs and the like. The output of the SIP typically comprises a plurality of images and defect reports (corresponding to defect report 30 in Fig. 1A) which may be displayed to the user on the screen of his workstation 32.

Typically, the system of Fig. 1B is operative in at least three scan scenarios or modes, as described in detail below: learn mode, design rule mode, and inspect mode. "Learn" mode typically comprises two sub-scenarios: "learn mixed" and "learn ref", which are preferably performed in an off-line inspection preparation step prior to processing a batch of panels. In the learn mode, information about panels to be inspected in a design rule and/or inspection mode are input to the system, or "learned" in order to serve as a reference for later processing.

As seen, two principal channels of input data (a snap data input channel from 50 and a CEL data input channel from 70) are typically received by SIP 40. Each of the two data input channels preferably comprises two subchannels: (i) a snap report sub-channel, designated herein "report snap", and a color-defect report sub-channel, designated herein "report color defect"; and (ii) a CEL report sub-channel, designated herein "report-CEL" and a defect report sub-channel, designated herein "report defect". These input data channels are supplied, as seen in Fig. 1B, by HIP 38, a preferred

embodiment of which is described in Applicant's above-referenced copending Israel Patent Application 131092.

The report defect sub-channel is preferably demultiplexed (unit 120) to obtain, as seen in Fig. 1B, four raw data arrays which are described in detail below: morphological feature (f), a hardware defect (hd), a binary CEL (c) and a color-CEL (cc). Generally, the channel task converter and splitter units 80 and 120 are operative to demultiplex raw data in order to separate out input data for use by the SIP 40. Demultiplexed data is termed herein "SIP raw data". The SIP raw data typically comprises several arrays of information. Each array is typically stored in a data structure termed herein a "ds_array" and described below in detail.

The following abbreviations are used to present demultiplexed data sources in a compact form in Fig. 1B and in the figures that follow:

C_0, C_1, C_2: denote data sources of type "Data structure array Cel" which relates to binary CELs. Each of the numerals correlates a data source to one of the three cameras in the preferred embodiment, 0, 1 and 2 respectively, from which it was obtained.

HD_0, HD_1, HD_2: denote data sources of type "Data structure array Defect" which relates to hardware defects, such as nick protrusion defects which are an unevenness found by the HIP in binary CELS. Each of the numerals correlates a data source to one of the three cameras in the preferred embodiment, 0, 1 and 2 respectively, from which it was obtained.

F_0, F_1, and F_2: denote data sources of type "Data structure array Feature" which relates to morphological features. Each of the numerals correlates to a data source to one of the three cameras in the preferred embodiment, 0, 1 and 2 respectively, from which it was obtained.

CC_0, CC_1, CC_2: denote data sources of type "Data structure array Color CEL" which relates to color CELs. Each of the numerals correlates a data source to one of the three cameras in the preferred embodiment, 0, 1 and 2 respectively, from which it was obtained..

CD_0, CD_1, CD_2: denote data sources of type "Data structure array Color Defect" which relates to a color defect, such as oxidation or a color surface defect, for example resulting from a chemical remnant or residue remaining on the

surface of a BGA. Each of the numerals correlates a data source to one of the three cameras in the preferred embodiment, 0, 1 and 2 respectively, from which it was obtained.

S_0, S_1, S_2: denote an SDD (small defect detector) data source which are small high contrast surface defects, such as scratches and pinholes. Each of the numerals correlates a data source to one of the three cameras in the preferred embodiment, 0, 1 and 2 respectively, from which it was obtained.

Qname: denotes the name of a data source queue of the type Window queue, which is a queue of windows created in response to inspection triggers that are waiting for image processing by SIP 40 (equivalent to analyzer 24 Fig. 1A).

WD: denotes a data source of type Width defects, which are defects in line widths in conductors and spaces between conductors in a BGA.

PIM mask: denotes a data source of type PIM services. A PIM is a point in map and Pim-services are collections of PIMs that are used to define a locations in an image being inspected in order to ensure correlation between a point being inspected and the application of appropriate inspection rules and methods for the particular location. PIM masks are zones that are drawn by the user, or that are created in a learn stage of inspection as described hereinbelow, and related to particular inspection execution sequences.

As shown in Fig. 1B, one of the inputs to the SIP 40 is user defined configuration data 180 which typically specifies a SIP scenario, for example one of learn, design rule or inspect mode scenarios.

Execution Scenarios for BGA Inspection with SIP

The SIP is a multipurpose flexible image processing unit that may be readily adapted to inspection needs of various objects, and may be operated in the context of various inspection scenarios. For the purpose of illustration, various inspection scenarios under which the SIP may be operated are described with reference to the inspection of BGA panels 36.

Preferred scan scenarios, illustrated with respect to BGA inspection, are represented graphically herein as execution graphs, and are not intended to be limiting to BGA inspection, but rather representative of suitable objects that may be inspected

using the SIP. Each of the preferred inspection scenarios or modes is operative to perform a different inspection functionality which are generally as follows:

א. Design rule scenario in which a panel is inspected with reference to a set of general design rules. Inspection in the design rule scenario is intended to provide a fast view of hardware defects (nicks and protrusions) and of minimum line width violations relating to conductor widths on BGAs and minimum space violations relating to the minimum space between conductors on BGAs. Execution of a design rule scenario will be described in greater detail with reference to Figs. 2 and 3;

ב. Learn scenario in which the contents and structure of the design of a panel is learned in order to generate a reference for use during the inspection of a batch of like panels. Learn scenario includes learning general attributes of a panel of a reference panel, such as actual line widths along each section of a conductor, and learning the location of areas of particular interest in the panel as generally described hereinabove with reference to Fig. 1A. Execution of a learn scenario will be described in greater detail with reference to Figs. 4 – 10;

ג. Inspect scenario in which a panel is inspected to determine defects with reference to a golden panel having known desired attributes. Inspect scenario is the scenario that is typically used to inspect batches of BGAs for manufacturing and process defects. Execution of an inspect scenario will be described in greater detail with reference to, Figs. 11 - 16;

It is appreciated that each of the SIP execution graphs is connected to at least one of the data input channels 50 and 70 of Fig. 1B. The input of channel data from channels 50 and 70 is dynamic and adapted to the particular needs of each inspection scenario. Data in each of channels 50 and 70 may preferably be provided in a file format for off-line inspection, or TCP/IP (Transport Control Protocol/Internet Protocol) format. Configuration variables are typically provided in advance of performing an inspection scenario to determine if the data from input channels 50 and 70 is from a FILE channel or from a TCP/IP channel.

Scan scenarios are generally described herein in the context of SIP configuration files, each of which defines a collection of inspection functionalities that

are performed in the context of the respective scenario. Whenever applicable, the name of a configuration file containing an execution graph is stipulated herein. The following set of preferred scan scenarios is now described in more detail:

A preferred SIP scenario entitled "design rule" (having a configuration file denoted dr.config) is now described with reference to Fig. 2.

This scan scenario is intended to provide fast viewing of hardware defects and of minimum line width and minimum space violations, and may be performed either from a file or on-line as decided by a user prior to execution of the scenario. This scenario is typically run on a panel being considered for use during the learn scenario described in greater detail below.

Typically, the design rule scenario requires minimum setup time, provides an initial rough indication of whether the quality of the image of a panel to be inspected and quality of the inspected panel are sufficiently acceptable to enable a particular panel to be used as a golden panel from which a comparison reference can be constructed for use in an inspect scenario. The scenario preferably identifies all predetermined hardware (nick and protrusion, CEL on Skeleton, etc.) defects, all color surface defects, all minimum line width violations on conductor regions in BGA panels, and minimum space violations between conductors in BGA panels, and outputs a report of all such as defects.

The aforementioned defects are preferably identified without correlation to any external reference. It is readily appreciated that if a BGA panel exhibits a preponderance of the previous defects, it is not suitable to be used as a reference. However, the fact that a BGA panel passes all of the above tests, does not necessarily ensure that the panel can be used as a suitable reference, and it is preferably inspected to ensure that all of the features exist, and that the features are properly positioned on the panel in accordance with design specifications.

As seen in Fig. 2, user defined design rule configuration data is input into SIP 40. Such data typically comprises the minimum used distance between conductors, and the minimum line width thickness of conductors. Input image data regarding a panel 36 is received from snap channel 50 and CEL channel 70, and is demultiplexed into the following image information data: color defect data for each camera, binary CEL data, and hardware defect (nick and protrusion) data.

Suspected defects for the image data input for each camera 0 – 2 is identified by an identifier of defects 210, an appropriate inspection method is identified for each suspected defect, is packaged into a window including image data for the region of suspected defect, and is operated by a data structure storing window 220 operative to package data methods and place them into a data queue for processing. Defect report generator 230 (analyzer 15 in Fig. 1A) inspects the image data in the window according to the identified method. If an actual defect is found, then the defect is reported in defect report 240.

Reference is made to Fig. 3 which is a sub-execution graph illustrating the flow of image data in and between identifier of defects 210 and data structure storing window 220 in Fig. 2. As seen, binary CELs from channel 70 are provided to a detector of defects 250 which is operative to detect defects correlating to too narrow conductors or lines and too narrow spaces between conductors or lines. An output of "too narrow" defects is provided to a width defects data report (WD Data Structure) 252, and report 252 is provided to a merger unit 254. In addition to report 252, merger unit receives a report of color surface defects from channel 50, a report of hardware defects from channel 70, and input data from GUI for off-line definitions of defect analysis mask 256. Merger unit 254 merges the reports, filters out reports that are found inside mask areas 258 and outputs a unified data report 260 in the form of a defect data queue which is queued for downstream processing. Mask areas 258 are generated through a user interface.

Once a BGA panel is inspected in the design rule scenario and is determined to be suitable to serve as a reference against which inspected panels may be compared, it is necessary to perform a learn scenario of inspection in which the particular structure of the panel is learned and stored in memory. Information about the learned panel is subsequently used in the inspection scenario discussed hereinbelow with reference to Figs. 11 -16, as a reference against which inspected panels may be compared to determine the presence of defects.

The learn scenario is now described with reference to Figs. 4 - 10.

Reference is made to Fig. 4 which shows a simplified functional block diagram of a learn mode selected by a user. User defined learn configuration data 180 is input, preferably via a general user interface 170 (Fig. 1B). A panel 36 is scanned,

and image data is generated. The following image data from each one of the cameras is used: color CEL data 130, 134 and 138; binary CEL data 140, 144 and 148; hardware (nick and protrusion) defect data 150, 154 and 158 and morphological feature data 160, 164 and 168.

Learning is typically performed on image data arriving from each separate camera, by applying three execution graphs in this order:

"Learn mixed", the particulars of which are contained in a configuration file called learn.mixed.config, and is described with reference to Figs. 5 – 8;

"Learn reference" the particulars of which are contained in a configuration file called learn.ref.config, and is described with reference to Fig. 9 and 10;

At the end of the learn scenario, scenario specific output data is received and stored for use during the inspection scenario.

Typically, each of the scan scenarios receives raw data from an off line FILE channel. The raw data is extracted from a DMA (direct memory access) type data input channel with a grabber program activated by the application. The program reads information from DMA channels and writes the raw data into files. The learn scenarios then read the data from those files. A preferred embodiment of each learn execution graph (or scenario) is now described in detail.

Reference is now made to Fig. 5, which shows an execution graph 300 for the learn mixed scenario. The configuration of the learn mixed scenario is preferably stored in a configuration file.

In the learn mixed scenario, a window is defined around each of the features characterizing a mask zone. The execution graph for the learn mixed scenario, typically uses each of the windows which are defined by a user, for example as defined through a GUI in an off-line process prior to running the learn mixed scenario. The input files into PIM defects filter 310 are thus user defined windows. PIM defects filter 310 outputs to a defects handler 320 which is operative to generate windows of interest in the form of SIP defects reference and defect windows reference windows 330, that are later used in the inspect scenario.

User defined windows that may be defined in the learn mixed scenario with respect to BGA type inspection comprise one or more of the following types of zones:

- target –is an area that has a feature of undefined shape that a user desires to ensure that particular shape is maintained in inspected BGAs;
- balls –is an area characterized by numerous circular shaped pads on a BGA;
- bonding area –is an area which is characterized by numerous bonding pads or power lines on a BGA;
- cavity –is an area that needs to be clear of any morphological features or CELs;
- chip area –is an area on a BGA where a chip is to be located.

The type of functions that are executed for each window type are defined in a definition file.

The execution graph of Fig. 5 receives the following demultiplexed image data:

a. color CELs from each of cameras 0 – 2, reference numerals 130, 134 and 138

b. binary CELs from each of cameras 0 – 2, reference numerals 140, 144 and 148

c. hardware defects from each of cameras 0 – 2, reference numerals 150, 154 and 158

d. morphological features from each of cameras 0 – 2, reference numerals 160, 164 and 168

Demultiplexed data is provided to execution units 340 which are associated with each camera and which define the following execution sequence:

Reference is made to Fig. 6 which is a preferred sub-execution graph illustrating the execution sequence performed in each of execution units 340. The execution graph of Fig. 6, which relates to each execution unit 340, deals with raw data coming from one camera. The Task Packer / Cel Packer task 342 packs raw image information data into inspection windows according to user defined windows 344, which are defined in the general user interface of workstation 32 (Fig. 1B) and passed to task packer 342 as an event service 345. Each inspection window comprises raw image information data and identifies a test function in response to the type of user defined

window 344. The window is pushed into a queue of windows 346 by the task packer 342 and is taken from the queue by the test manager 348. The test manager pops a window from the queue, executes the function that is associated with this window and pushes the window into one of the output queues according to the forward destination of the window. If test manager 348 is able to complete the test on the window, the window is forwarded to a unified reference queue 350, and then joined in the SIP defects handler 320 (Fig. 5).

If the test manager 348 is unable to complete the test function based solely on the raw image information received for a single camera, then the window is pushed into unresolved queue 360. This may occur, for example, if a user defined is located so that it is not entirely imaged by a single sensor.

Returning now to Fig. 5, it is seen that unresolved queues 360 from each of execution units 340 are pushed in the appropriate order into a multi slice manager 370 which is operative to join each of the unresolved windows with the appropriate window to generate a joined inspection upon which an inspection task may be performed.

Multi-slice manager allocates information according to the following sequence, in order to provide one or more of the functions detailed in Fig. 8:

All of the fundamental image data for corresponding unresolved windows coming from neighboring slices, acquired by neighboring image sensors, is unified into a unified window, and an appropriate function is applied to the unified window according to its type.

If incoming image data comprises data for balls, then the data is provided to a balls reference queue 380. In the appropriate order, a balls measurement unit 382 receives the data measures the balls in the region and outputs a data file SIP balls measurement data 384, which indicates the nominal balls measurement. A preferred method for measuring balls is described in Appendix J.

If incoming image data comprises data for bonding areas, then the data is provided to a bonding areas reference queue 390. In the appropriate order, a mask extractor 392 receives the data and extracts from the data masks which surround each various inspection regions in the window, such as bonding pads and power lines in the context of BGA inspection, and outputs a data file called polylines reference mask and

stable features reference 394. A preferred method for identifying regions of interest and distinguishing them from other regions is described in Appendix C

If the incoming data for a slice does not contain either a ball nor requires mask extraction, the information is provided to a unified queue 398, along with any balls measurements and extracted masks. The unified queue is provided to defects handler 320 for additional processing.

It is readily appreciated that the functionality and sub-execution graph of test manager 348 (Fig. 6) is substantially similar with respect to raw data relating to a single camera as is the functionality performed by multi-slice manage 370 with respect to a combination of slices received from multiple camera inputs.

The functions which are executed in each inspection window in the learn scenario are defined in a file. The table of Figs. 7A - 7B lists window types and their corresponding test function (or list of functions when the window is attached to a composite function), and provides a brief description of the functionality. The table also describes the action of the functions on the window and the return status, including the destination string that determines to which of the output queues the window will be sent.

Both unified queues 350 and 398 go to SIP defect handler task which writes defects and produces a top down reference. A top down reference is a reference that ultimately functions as a spatial area of interest during the inspect scenario, as described with reference to Fig. 1A. A function that is attached to each top down window is performed at each spatial location associated with the top down window on the basis of location. The following functions may be executed before a top down reference is created:

a. Strip balls leaves only circle shaped data structure in the top down window to define spatial regions of interest around each circle shaped data structure.

b. Strip bonding returns a sub window directive that instructs the task to write a different reference window for each sub window which is associated with a respective bonding pad feature in the window.

c. Strip target does nothing in the top down window.

d. nop. Does nothing.

It is noted that in the learn stage, parts of the panel which are known ahead of time to be of particular interest from an inspection point of view, for example balls area, pad areas and target areas in a BGA inspection application, may be identified so that each part can serve as a trigger for additional inspection in the inspection scenario described hereinbelow.

When a window is constructed by task packer 342, the data stored inside each window is whatever data is necessary to model the part in the window and to perform each of the inspection functions that is necessary to inspect the content of the window with a robust inspection method. For example, in BGA inspection, for balls areas a list of each ball in the area, including center point and radius is stored. For target areas, a compressed polygon describing the boundary of the target is typically stored.

Reference is now made to Fig. 9 which is a preferred execution graph for the learn reference scan scenario. In the learn reference scenario, general reference data is learned for an entire panel, without correlation to any particular spatial region interest. The execution graph of Fig. 9 is operative in BGA inspection to define reference data for the following attributes of a BGA:

a. line widths data, which is the nominal width of conductors and spaces between conductors on a BGA;

b. Registration data, which is set of morphological features in the image of a reference object along with an identification of features determined to be stable in the learn scenario;

c. CEL data which is data on the location of CELs in the reference image;

d. Feature data which is data on the type and location of features in the reference image.

The execution graph of Fig. 9 uses the following input data:

a. Demultiplexed raw image data for morphological features;

b. Demultiplexed raw image data for CELs.

c. Polylines reference data developed in the learn mixed scan scenario, described with reference to Fig. 5;

d. Disable masks data, which define zones which serve as masks in which line width computation are not performed, for example in BGA inspection, "target", "balls" and "mask" zones.

e. Stable features reference, in which defines areas in which features are deemed to be stable, namely that they have a stable location..

The execution graph of Fig. 9 generates the following outputs for the learn reference scenario:

a. Rectangles around areas on which line width can not be computed. The information is stored in a learn line width disqualified window file;

b. Line width reference which, in BGA inspection, stores information about the widths of conductors and spaces between conductors;

c. Reference features for registration in a reference feature file;

d. Raw reference of CELs and features in CEL and morphological feature files, each of which is indexed and comprises reference data relating the feature information to a camera from which obtained.

As seen in the execution graph of Fig. 9, demultiplexed raw image data for binary CELs from each camera 140, 144 and 148 and for morphological features from each camera 160, 164 and 168 is provided to a SIP composite task unit 400. Each composite task unit 400 is associated with a camera. The execution graph with respect to composite task unit 400 is described in detail with reference to Fig. 10.

Additionally, demultiplexed raw data information for morphological features, preferably information for each camera although information from only one of the cameras may be used, is provided to a task learn registration unit 420, which receives stable features data 422 which are features having a known and stable location on panel 36. Stable feature data is developed in the learn mixed scenario.

Learn registration unit 420 is operative to learn the type and location of stable features on panel 36 (Fig. 1B) and to generate a reference file 424 (FILE REF_FEATURES) that is used in the inspect scenario as a reference for determining excess and missing features, and for registering an inspected panel against the reference panel.

Additionally, demultiplexed raw data information for CELs , preferably information for each camera although information from only one of the cameras may be

used, is provided to a task learn line width unit 430, which receives a mask of polylines data disable mask data 432 generated in the learn mixed scenario. Polyline data and disable mask data 432 is place in PIM, point in map, and 'MIM', something in map, references that relate line widths to regions on a panel 36 which is inspected in the learn reference scenario.

Learn line widths unit 430 is operative to learn line widths, correlate line widths to locations on panel 36 (Fig. 1B) and to generate a reference file 434 (FILE LEARN_LW_DISQAULIFIED_WINDOWS WIDTH REFERENCE FILE) that is used in the inspect scenario as a reference for determining whether a given calculated line width measurement falls within a predetermined range of acceptable measurements.

Reference is now made to Fig. 10 which is a sub-execution graph for the task performed in composite task unit 400. The sub-execution graph relates to the task performed on raw image data information for each camera.

The sub-execution graph comprises a geometric reference task unit 450 which is operative on demultiplexed raw image data input for CELs and for morphological features to compute a geometrical reference which identifies CELs and morphological features, correlates them to a location in the reference image and outputs reference files. A first file is CEL type and location reference 452, and second file is morphological feature type and location 454.

Reference is now made to Figs. 11 – 16 which show execution graphs of the inspect scenario, and describe inspection tests applied therein.

The following terminology is used in the ensuing description, and is helpful in understanding Figs. 11 – 16:

In the execution graph of Fig. 11, the following data sources are generated:

R_1: is a registration transformation between an image slice associated with the central cameras of reference and online images.

E_1, IM_1: Excess and illegal matches, respectively, for morphological features acquired by the central camera. An excess match is a morphological feature which exists in an online image but does not exist in the reference image. An illegal match is a morphological feature that exists in the online image, but whose coordinates match up to a different morphological feature in the reference image.

T_XX, where XX = 0,1,2: are transformations from coordinate system of camera XX into a reference aligned coordinate system, derived from R_1.

T_a2a: is a transformation from a reference aligned coordinate system into an online aligned coordinate system, derived from R_1.

Reference is now made to Fig. 11, which is an execution graph for a registration and determination of excess and missing features task. The execution graph of Fig. 11 is performed during an inspection scenario in registration generator 18 of Fig. 1A.

This execution graph gets raw image data, comprising color CELs 130, 134 and 138, binary CELs 140, 144 and 148, hardware defects 150, 154 and 158, morphological features 160, 164 and 168, and color defects90 from each of the cameras in scanner 34 (Fig. 1B; for the sake of simplicity only data from camera 0 is shown).

The registration task unit 510 is, in the illustrated embodiment, for simplicity, operative to register features of camera 1, which is the central camera. Registration task unit 510 receives features 164, along with reference features 520 from memory, and is operative to produce an output of a registration transform 530 with respect to camera 1, an excess morphological feature report 532 with respect to camera 1, and an illegal matches report 534 with respect to camera 1. A preferred method for generating reports 530, 532 and 534 is described in detail with reference to Appendix E.

The transformation computed is split by a transform splitter unit 540 into three camera-to-reference aligned transformations, T_0, T_1 and T_2, one for each camera, and to a transformation between the two aligned coordinate systems T_A2A, which is a transform between the reference data and online data from a panel being inspected. All of the data sources, raw image data and computed data with respect to registration transformation, excess and missing features, and illegal matches are the input to inspect task unit 550, the execution graph with respect to which is shown in detail in Fig. 12.

Reference is made to Fig. 12 which is an execution graph for inspection unit 550. Inspection unit 550 receives inputs as slices, one for each camera, as described with respect to the execution graph shown in Fig. 11. Inspect task unit 550, comprises first composite task inspection task units 560 which are operative on the outboard image slices, as which are provided by cameras 0 and 2, and a second

inspection task unit 600 which is operative on the central image slice which is provided by camera 1.

Reference is now made to Fig. 13 which is an execution graph showing the functionality of outboard composite task inspection units 560. Each outboard composite task inspection unit 560 comprises a task packer unit 570, which corresponds to task packer / window of interest generator in Fig. 1A, and which is operative to receive inputs as described with reference to Fig. 12, and in addition a top down reference. The top down reference comprises spatial areas of interest generated, for example by a spatial area of interest identifier during the learn scenario as described. For each inspection trigger, for example a top down window, or hardware defect, or morphological feature and which are not inside any of the top-down windows, that require additional inspection, task packer 570 is operative to generate an inspection window that identifies an inspection method and includes the used image data in order to perform the inspection method. The inspection window is pushed into an inspection window queue 572. A test manager 580 is operative to take inspection windows from inspection window queue 572 and apply the test method to the image data in the window. It is readily appreciated that test manager is operative on data with respect to a single camera. Thus, features, regions and top-down windows that extend outside the camera boundaries, or which cover regions that are overlapped by more than one camera are unresolvable, and are pushed into an unresolved queue 582, while fully resolved defects are pushed into defects queue 584.

In parallel to the operation of task packer 570 and test manager 580, two additional functional test units a CEL watchdog unit 590 and side defects watchdog unit 595 are operative in composite task inspection unit 560. CEL watchdog unit 590 is operative to receive binary CEL and transform inputs along with a PIM CELs reference which defines regions that are to be free of Cels. CEL watchdog unit 590 watches CELs to ensure that no CELs are located in areas which have been designated by PIM CEL reference to be free of CELs. If a CEL is found in a regions designated to be free of CELs, a defect report is pushed into defect queue 584.

Side defect watchdog 595 operates as a filter for hardware defects and color defects. PIM Main defines a masked area. If a color defect or hardware defect is located in a region masked by PIM Main, no defect report is issued. If a color defect or

hardware defect is located outside of a masked area, then a defect report is pushed into defect queue 584.

It is noted that hardware defect data source typically provides inputs to both side defect watchdog 595 and task packer 570, and a switch P. Typically switch P is turned off so that data from the hardware defect data source does not arrive to side defect watchdog 595, so that defect reports are received only after hardware defects have been run through further inspection facilities provided by task packer 570 and test manager 580. If switch P is on, then a double report is issued which is useful for control and testing of the effectiveness of test manager 580 in filtering real and false defects.

Reference is now made to Fig. 14 which is an execution graph showing the functionality of inboard composite task inspection unit 600, which is provided with respect to image data from the central camera, camera 1. The operation of inboard composite task inspection unit 600 is similar to the operation of outboard composite task inspection units 560, however it is operational to process additional information arriving with respect to line width defects as shown with respect to Fig. 11.

A line width inspect task unit 610, which corresponds to line width calculator in Fig. 1A, is provided to calculate line widths in online images of a panel under inspection 36 (Fig. 1B). Line width inspect task unit 610 is operative to receive line width reference data 612, and in the embodiment described, registration data and binary CEL data for the central camera, designated camera 1, as shown. Line width inspect task unit 610 applies the input registration data to the binary CELs, and compares the result with a reference. If a line width defect is detected, a line width defect report 614 is issued.

A task packer unit 570, which corresponds to task packer / window of interest generator 12 of Fig. 1A, and which is operative to receive inputs as described with reference to Fig. 12, and in addition a top down reference. It is noted that in addition to the inputs received by the task packer unit 570 shown in Fig. 14, additional inputs are received, namely width defect report 614, excess missing report for the central camera E_1, and illegal match defect report for the central camera IM_1.

For each inspection trigger, for example a top down window, hardware defect, or morphological feature, or line width defect, or excess/missing or illegal match, and which are not inside any of the top-down windows, which necessitates

additional inspection, task packer 570 is operative to generate an inspection window that identifies an inspection method and includes the used image data in order to perform the method. The inspection window is pushed into an inspection window queue 572. A test manager 580 is operative to take inspection windows from inspection window queue 572 and apply the test method to the image data in the window. It is readily appreciated that test manager 580 is operative on data with respect to a single camera. Thus, features, defects, and top-down windows that extend beyond the camera boundaries, or which cover regions that are overlapped by more than one camera, are deemed unresolvable, and are pushed into an unresolved queue 582, while fully resolved defects are pushed into defects queue 584.

In parallel to the operation of task packer 570 and test manager 580, a CEL watchdog and defect watchdog unit 620 is provided. Watchdog unit 620 operates as a filter for line width defects, color defects, hardware defects, excess/missing defects and illegal match defects. PIM Main defines masked areas, and transform information for the central camera T_1 is provided for registration. If a line width defect, color defect, hardware defect, excess/missing defect or illegal match defect is located in a region masked by PIM Main, no defect report is issued. If a line width defect, color defect, hardware defect, excess/missing defect or illegal match defect is located outside of a masked area, then a defect report is pushed into defect queue 584.

It is noted that hardware defect data source, excess missing data source E_1, illegal match data source IM_1 and line width data source 614 each provide inputs to both to watchdog 620 and task packer 570, and include a switch P. Typically switch P is turned off so that data from the above data sources does not arrive to watchdog 620. When turned off, defect reports are received only after hardware defects have been run through further inspection facilities provided by task packer 570 and test manager 580. If switch P is turned on, then a double report is issued which is useful for control and testing of the effectiveness of test manager 580 in filtering real and false defects.

It is noted that inspection windows generated by task packer 570 may be of one of two classes of windows:

(1) predefined (top down) inspection windows taken from the input reference file, and

(2) windows around defect (hardware, excess/missing, line width) and feature triggers that are not within the top down windows.

In a preferred embodiment of the present invention, top down windows are provided for features for which a robust model has been created. For example, in BGA inspection, robust models have been generated for balls, bonding pads, targets and cavities. Preferably the models are dynamic and are capable of intelligently identifying shapes that may not precisely fit a predefined model, but rather that deviate in various ways, according to a generalized inspection routine.

Preferably, inspection windows are provided for features that have an unknown shape or a shape that is not predefined according to a model. For such windows, inspection is preferably performed by comparing a general method that directly compares the shape in the online image of the panel being inspected to the shape of a corresponding feature existing in the corresponding region of a reference. Preferably a comparison mode employing comparison of edge contours is employed as described below.

It is readily appreciated that by combining region dependent localized modeling functions and comparison to reference functions, inspection capabilities are generally optimized to enable detection of defects in objects having predefined characteristics and objects that do not have predefined characteristics.

Functions that are executed for each top-down inspection window type are defined and stored in a file. The table of Figs. 15A - 15C lists each window type and its corresponding test function (or list of functions when the window is attached to a composite function). The table of Figs. 15A - 15C also describes the action of the functions on the window and the return status, including the destination string that determines to which of the output queues the window will be sent.

Referring again to the execution graph shown in Fig. 12, unresolved queues 582 from all three cameras are pushed to task multi slice manager 630. This taskcollects all unresolved windows having the same identification tag and combines inspection windows from joining slices, executes their function and pushes defects into a unified defect queue 640. The table of Fig. 16 lists possible functions executed the multi-slice manage 630 in the course of an inspect scenario.

Defect handler task unit 650 preferably is operative to receive alignment transform data to register between a reference image and an online image of panel being inspected 36 (Fig. 1B) and a unified defect report from the unified defect queue 640. Defect handler task 650 outputs a list of defects 660, that is extracted from windows in the unified defect queue 640. Optionally, if instructed in a configuration file, defect task handler 650 outputs file of defective windows, accompanied by its relevant data, for example CELs, color CELs, morphological features, defects in response the type of window. The file of defective windows is useful, for example to debug operation of defect handler 650.

Windowing Inspection Environment

It is appreciated that SIP based image processing assumes that "interesting" occurrences, from an image processing point of view, typically happen within rectangular Regions Of Interest (ROI) or windows that occupy only a relatively small portion of the surface area of an object under inspection. Accordingly, SIP inspection comprises at least some inspection routines that are preferably limited to areas of interest. This avoids having to perform complicated inspection methods over extended images in order to identify inspectable characteristics that are believed to located inside a region of interest that surrounds an identifiable inspection trigger.

In order to accelerate computation and allow more complex image understanding, preferably a three part image processing process is employed.

In the first stage of the process, predefined spatial regions of interest, e.g. top down windows, are defined as additional inspection triggers and are stored in an off line memory. The top down windows define reference image information for the region of interest included in the top down window, and identify an image processing function, or sequence of image processing functions, that are to be applied to analyze regions of interest at the same spatial location as the top down window in online images of objects under inspection.

In the second stage of the process, a fast "quick and dirty", preferably (but necessarily limited to) hardware based image processing is used to identify additional inspection triggers and to generate parsed fundamental image data for online

images of an entire object under inspection. The parsed fundamental image data is passed to the SIP for SIP image processing and analysis.

In the third stage a three part image processing execution is preferably performed. First, computational tasks are globally applied to the parsed fundamental image data to compute various computational information about the image, for example computation of line widths, excess features, missing features, illegally matched features and registration transformation. Second, windows of interest are defined around additional inspection triggers, for example predetermined features and candidate defects output from the second stage and the first part of the third stage. Third, windows of additional inspection are received for top-down windows created in the first stage, and for additional inspection triggers located outside top down windows, windows of additional inspection are generated.

Additional windows, including both top-down windows and windows defined around additional inspection triggers, are dynamically packed to comprise fundamental image data from an online image of an object under inspection for the region of interest, identification of an appropriate image processing function, or sequence of image processing functions, necessary to inspect the region of interest and any reference data that is needed by the image processing function for image processing. The image processing function is preferably chosen in response to the type of region of interest in order to provide optimum image inspection for the type of region or inspection trigger in the inspection window.

In the aforementioned preferred architecture, intelligent (and resource demanding) image processing computations may be applied only to specified inspection windows on particular regions of interest, rather than over the entire object.

A "window inspection trigger" which triggers additional inspection is the center point of an ROI or window. As noted, window inspection triggers can be

a. Defects, e.g. defects detected by hardware (for example, nick/protrusion report) or by software (for example, excess/missing or line width reports),

b. Features, e.g. morphological events (islands and open-ends in an electrical circuit)

c. Top down triggers, e.g. a predefined locations that are defined by a user. For example – top down triggers may by rectangle ROIs drawn by a user around all

balls in a ball area, or alternatively, locations that are pre-specified as a result of running a learn scenario as described above .

A specification of the dimension of an inspection window that is to be created around a trigger is typically attached to each type of window trigger. In addition, window triggers preferably comprise data expected to be found in the window and a definition of the type of computation that is to be performed for data inside the window. Such definitions are typically specified in special configuration files.

Once fundamental image data is packed inside the window, the inspection window is ready to be processed. A typical processing function operates on an inspection window and uses the data inside the window . Sometimes, processing functions generate additional data that is appended to the window. Several processing functions on windows are described below.

A particular feature of a preferred embodiment of the invention is that different functions can be locally applied to regions of interest inside various windows. Accordingly, for example, for regions of interest associated with top down windows, intelligent and robust feature modeling functions may be applied to the region of interest without applying these functions indiscriminately to the entire image. Feature modeling in top down windows is particularly suited for features having a known shape that can be readily represented and inspected by an appropriate method, for example a ball pad geometrically defined as a two dimensional circle may be represented by a center point and radius.

For regions of interest associated with other non-top down reference window inspection triggers, a comparison based image processing function preferably may be applied. For example a comparison based image processing function is preferably suited to inspect features and candidate defects that are characterized by shape and dimensions which are not stable and which can not be readily modeled ahead of time.

In accordance with a preferred method of comparison, in the learn stage binary CELs are combined and converted into vector lines comprising directed components. The directed components represent edges or contours between morphological features and substrate in reference image, and are stored in memory for use during inspection of online images of objects under inspection. CELs and the

generation of vectorized line components to represent collections of CELs are described hereinbelow in greater detail with reference to Figs. 21A – 21D, and the CELs to vector comparison function is described hereinbelow in greater detail with reference to Fig.s 30A – 32B.

The following preferred steps are helpful in understanding operation of windowing

STEP I: A user defines a window in a reference object around a spatial area that typically comprises at least one unit of a predetermined model shape. It is expected that in inspected objects, such as panel 36 (Fig. 1B), at least one shape defined by the model will be situated in a spatial region corresponding to the region defined by the user in the reference object. An appropriate model shape is preferably chosen from a library of modeled geometric shapes which is stored off line. Preferably, each model shape is defined mathematically. For example, a circular "ball" 882 (Fig. 27), such as are typically found on BGAs, may be defined by a center point 884 and a radius.

STEP II: Attributes for the reference region are defined. For example, a user may set parameters for permitted deviation in an inspected image with respect to the location, size and other aspects of circular ball 882. Accordingly, center point 884 may be defined as having to be located within an acceptable distance from a predetermined coordinate, and the radius at various locations along the circumference may be defined as having to be within a predetermined range of acceptable radii. Other additional parameters may be set, such as whether any other features, shapes or image elements, such as a line, or edge, are permitted within a region surrounding the modeled shape. Steps I and II are preferably performed in a learn stage as described hereinabove.

STEP III: Images of an inspected object are acquired in an inspection stage. Preferably, images are acquired using apparatus in accordance with a preferred embodiment of the present invention as described herein, and are preprocessed to produce one or more forms of fundamental image data, such as CELs, color CELs, morphological feature information and other information about the image.

STEP IV: In an online image of an object which is inspected, additional inspection regions are isolated. Each of the additional inspection regions

preferably spatially corresponds to one of the reference regions defined in Step II, or alternatively to a spatial region that surrounds some form of predetermined inspection trigger such as a defect or feature which the user defines during set-up as necessitating additional inspection.

Preferably, each of the additional inspection regions is packaged into an additional inspection window that comprises the stored shape model and fundamental image data, identifies one or more inspection functions that is to be performed on the window and specifies any user defined parameters that may be needed by the image processing function. For example, if the additional inspection region includes a ball 882, the window preferably includes all of the CELs for the region in the additional inspection window, any morphological features found therein, line width information and the like. Additionally the inspection window typically specifies the inspection functions that will operate on the ball, and user defined parameters for the ball, such as permitted deviation from the center from a desired location, and permitted ranges of radii for different sections of the circumference.

Accordingly, if an inspection window is generated for a region surrounding a candidate defect, a function operative to particularly identify the candidate defect, typically different than the function used to identify a modeled shape, is included in the additional inspection window.

STEP V: Image processing functions packaged with the additional inspection window are executed on the fundamental data in the window and evaluated according to the user defined parameters.

Preferably, if the additional inspection window includes a modeled shape, fundamental image data included in the additional inspection window is processed to generate a geometric object that represents a feature. It is appreciated that the fundamental image data is derived from an online image of the object being inspected in a spatial location correlating to the reference window. For example, if the feature being inspected in an additional inspection window is a ball 882, then all of the CELs in the inspection window may be combined to generate a line which represents the circumference of the ball. Once the geometric shape is generated, then the geometric shape in the window is compared to the model in the reference window.

It is appreciated while a library of many possible shapes may be created, it is not feasible to anticipate all possible shapes, particularly those shapes that are representative of defects, which are by their nature random.

Accordingly, in preferred embodiments of the invention there are provided functions that analyze the shape of geometrical features without reference to a mathematical model, but rather by direct comparison of a geometrical shape in the object being inspected at a particular location in the image, to the shape which is expected to be present at that location according to a reference. Direct comparison may be accomplished, for example, by generating and comparing geometrical shape representations, preferably by CEL to vector comparison functions as described hereinbelow, or by bit map comparison as is conventionally known in the art.

STEP VI: Reports are made to indicate the presence, absence and location of features and attributes in an inspection and whether any of the features are defective.

Preferred General Data Structure in SIP

Reference is now made to Fig. 17 which is a diagram of a preferred data structure of incoming data imported, from an image processor 5 (Fig. 1A) which may be used in the context of inspecting BGAs. The raw data which serves as input in Fig. 1 typically comprises a stream of data words organized as shown in Fig. 17. Typically, the SIP channel task converts incoming raw data into a data structure array element type representation in which an element is defined as one of the following: a binary CEL, a Color CEL, a Snap image, a morphological Feature, a Defect, or a Color defect. The structure of a data structure array and the structure of each element type in the data structure array are described in detail below.

In the preferred embodiment of the invention, raw data is received in a data structure as described in Fig. 17 and is split and converted into a data structure which can be included in a data structure array, for example by a data converter and splitter unit 120 shown in Fig. 1B.

A header for a typical report element, describing one row y of a scanned image, as received from one camera in scanner 34 (Fig. 1) is illustrated in Fig. 18. As shown, the header typically designates the type of the report element, for example, the

type of event that occurred in some row y, how many (cnt) times, and for which camera or slice.

One preferred typing scheme for identifying the type of report element is the following 3 bit scheme:

0 -- report no type

1 -- report color defect

2 -- report defect

3 -- report CEL which pertains to an edge

4 -- report snap

"Slice" refers to the number of the camera from among, say, 3 cameras which are employed to image the panel. Multiple camera and slice considerations are discussed herein below.

One preferred SIP data structure array is a data source called a "Data structure array Element". This data structure considers the data as being organized into lines, ordered by increasing Y coordinate. Each line contains zero or more data items (4 bytes in case of all SIP raw data items) of a particular type or element. The range of accessible lines comprises all lines from smallest line in an image up to the largest line in the image. An empty data source is identified when the following condition is met:

smallest line > largest line

When a data source is created, the value of the smallest line is set to 0, and the value of the largest line is set to -1. The data source specifies the number of the largest line in a scan, which is preferably greater than or equal to the coordinate of the largest line produced by the scanner.

Accordingly, each row in a scanned image corresponds to a row specified in a data structure. At any moment, the number of rows between the smallest line and the largest line in a data structure correspond to the scanned image rows which are ready for processing.

Each element in a data structure array typically has only one producer and at least one consumer. Lines can be added to the data source only by the producer and only if the y coordinate of the added line is greater than largest line in the data

source. When a line is added, all lines between largest line and the new line (not including the new line) are added as empty lines, each of which contains zero data items. If the new line is not empty, the new line with all its elements is added and the largest line is updated to be equal to the new total number of lines. A producer of a data source needs to ensure that at the end of scan, the largest line of the data source will be equal to largest line in the scan. It is appreciated that the largest line may be empty.

Preferred Data and Queue Management in SIP

Typically, the SIP comprises a plurality of tasks which are managed by a SIP manager that takes care of the flow of data from a front end image processor such as image processor 5 in Fig. 1, and between the tasks. The SIP manager takes care of the synchronization between reading data from external raw data channels and performing computations. The manager typically does not care about the specific types of data in the system nor about the sort of computations that are performed by the various computational tasks.

The SIP manager typically manages data and computation flow between tasks, manages input and output of data from and to external channels, provides a uniform error and logging mechanism, and provides timing and memory consumption statistics.

Basic building blocks of the SIP comprise data sources, tasks, execution graphs, and configuration file, all of which are now described:

1. Data sources

All data used by more than one computational unit is typically in the form of a data source. Each data source preferably has producers and consumers (tasks or channels, see description below). The producers generate the data source and the consumers use data from the data source. Each consumer is responsible for notifying the data source of data that is no longer needed by that consumer. Data not needed by any of the consumers can be safely removed from the data source to free up memory space and resources for more incoming data. It is the task of the SIP manager to remove data to free up space.

2 Tasks

Preferably a task is a computational unit that is operative to perform some computation on at least one input data source. The result of the computation is pushed into at least one output data source. An individual task is preferably not involved with data handling (for example - overflow of internal buffers or synchronization between data and computation), but works on some data. Data handling issues are typically handled by the SIP manager.

A channel task is responsible for the connection of the SIP as a process running on a computer to the outside world, for example operating on input data to demultiplex the input data and convert it into a format that is usable by the SIP. Each data channel is typically attached to a physical channel (DMA, TCP/IP or file channel). Input channels convert incoming data into data sources. Output channels are attached to data sources and they write the data source (as soon as it is updated) into the designated channel. Use of a file channel allows easy simulation of a scan without the need to actually connect to a DMA or to a TCP/IP channel. Incoming data may simply be injected from a file containing data in the same format as data arriving from DMA or TCP/IP channels.

Some tasks, notably inspection tasks, call functions that perform image processing computations on data.

3 Execution graphs.

A given execution graph (also termed herein a "scan scenario") is defined by the set of data sources and tasks participating in the scenario.

Reference is now made to Fig. 19, from which it is noted that a scan scenario may be generally represented by a graph as shown therein.

In the scenario of Fig. 19, the SIP is connected to two input channels and one output channel. There is one DMA channel 600 producing data source 2 and one TCP/IP channel 610 producing data source 1. There are three computational units (tasks): task 1 which uses data source 2 and data source 1 and produces data source 3; task 2 which uses data source 1 and does not produce any data source; and task 3 which

uses data source 3 and produces data source 4. Data source 4 feeds into a TCP/IP channel (CHANNEL 3) and is written into it as soon as it is updated by task 3.

Preferred runtime behavior of an execution graph, including a description of how data may be processed during scan is described in Appendix A.

4 Configuration files.

Execution graphs and all the parameters employed by channels, tasks and data sources are typically specified in configuration files. This allows many different scan scenarios to be run without needing to recompile or relink.

Inclusion of scan scenarios in configuration files makes expanding and changing the SIP relatively easy. From a software development point of view, one of the most time-consuming aspects of a SIP typically includes data handling and synchronization between data acquisition and computation which, however, may be very similar or even identical for many AOI (automated optical inspection) applications. Preferably, the SIP is constructed such that it is easy to add tasks, and easy to add data sources that are instantiation of built-in templated data sources. Accordingly, to add new data sources or task channels, a suitable class of data sources or tasks is derived from a corresponding base class of data sources and tasks already in existence. As much as possible of the work typically is thereby transformed to the base class (by extensive use of a template pattern).

Reference is now made to Fig. 20 which is an execution graph for two processes. A given execution graph can typically be processed by more than one SIP. To do this, the execution graph may be divided into several execution graphs, each running on a different SIP. Data sources produced by a task in one SIP may be used by the task in another SIP. As seen in Fig. 19, a data source is written into a TCP/IP channel by the first SIP and read from the TCP/IP channel by the second SIP. Minimal cross talk provides short computation time between the two SIP processes.

Principal Forms of Fundamental Image Data – CELs and Events

The following section describes preferred methods by which data is represented in data structures in SIP for fundamental image data, with respect to a preferred embodiment of the invention employed in the inspection of BGAs.

Binary CEL Representation

In the following discussion the following definition of a CEL (Contour Element) is used: a CEL is a single directed line defined within a pixel. The location and orientation of a CEL in a pixel is calculated as a function of the difference in gradients of reflection intensity as registered on neighboring pixels. Collections of CELs generally define boundaries between two populations within a pattern formed by a binary image (namely an image that comprises only two populations), and are operative to define the boundary to with a sub-pixel accuracy. If each CEL within an image is oriented in a direction, then the orientation of each CEL is defined so that the "white" area in a binary image comprising only white and black areas always lies to the left of an oriented, or directed, CEL.

CELs are calculated as a function of the zero crossing point of DoGs (Difference of Gaussians) as described in U.S. Patents 5,774,572 and 5,774,573 to Caspi et al., incorporated herein by reference. Other suitable edge detection techniques are described in: C. Gonzalez and P. Wintz, *Digital Image Processing*, Addison Wesley, Reading, MA, 1987.

In the present invention, the CELs are preferably calculated for pattern edges in the Red band of an RGB image. "White" area is the area in which the values of DoGs are negative.

Reference is now made to Fig. 21A which illustrates a pixel 710 having therein a CEL 720. Pixel 710, is, for example, a pixel in an image of an object to be inspected 4 (Fig. 1A), having one corner located at an origin (X,Y) and a diagonal corner located (X+1, Y+1). CEL 720 is located in pixel 710. Pixel 710 straddles a "white" area 730 and a "black" area 740. As shown in the figure, CEL 720 is vectorized so that white area 730 lies to the left of CEL 720.

A preferred 24 bit data structure for representing CELs, as elements in a CEL data structure array for use in SIP comprises the following fields:

x (12 bits) is the X coordinate of a CEL. Note that inasmuch as each line in an image are grouped into a data array, the y coordinate of a CEL is specified from the line in a CEL data array in which a CEL is contained.

direction (1 bit) and edge code –(3 bits) define the orientation and direction of a CEL.

last or "L" –(4 bits) defines the location of the last endpoint of a CEL along an edge of a pixel. An endpoint is the point along the edge of a pixel where intersected by a CEL.

first or "F" –(4 bits) defines the location of the first endpoint of a CEL along an edge of the pixel.

Reference is now made to Figs. 22A - 22F which depict possible CEL orientations corresponding to edge code values of 0 to 5 respectively, in which the direction value is 0, and are generally self-explanatory. Accordingly, an edge code value 0 is assigned to the CEL of Fig. 22A, an edge code value 1 is assigned to the CEL of Fig. 22B, an edge code value 2 is assigned to the CEL of Fig. 22C, an edge code value 3 is assigned to the CEL of Fig. 22D, an edge code value 4 is assigned to the CEL of Fig. 22E, and an edge code value 5 is assigned to the CEL of Fig. 22F.

Reference is made to Figs. 22G and 22H which illustrate edge code value of 6 in which the direction value is 0. A saddle occurs when two CELs exist in a pixel as shown in Fig. 22H. In the case of a saddle, individual CEL information is represented in a manner similar to Fig. 22F. An edge code value of 6 nevertheless differs from an edge code of 5 in that during a later stage of combining CELs in order to calculate a contour representing an edge, an edge code of 6 enables reconstruction of the saddle based on an evaluation of neighboring pixels.Typically, code values are assigned to represent each of the situations of Figs. 22A – 22F, and 22H and an additional "illegal" code value is reserved to indicate skeleton events.

Reference is now made to Figs. 23A - 23F which depict possible CEL orientations corresponding to edge code values of 0 to 5 respectively, in which the direction value is 1, and are generally self-explanatory. Figs. 23G and 23H represent a saddle situation existing when direction value is 1, and are generally analogous to Fig. 22G and 22H, except that the relationship of “white” areas and “black” areas is reversed.

Once an edge code and direction code are defined, the orientation of a CEL with respect to edges of pixel are known, and the first and last points of a CEL can also be defined. The first point is the point denoted by "F" in Figs. 22A - 23H while the last point is the point denoted by "L".

The first field and last field of the CEL type preferably denote the locations of the first point and last point on a cyclic coordinate system. This coordinate system is best seen in Fig. 23I.

Preferably, each edge of a CEL is divided into units of 1/16 for representation in 4 bits. Thus in the example of Fig. 23I, the value of "F" is 14 and the value of "L" is zero.

Vectorized Representation of Edges of Features

Reference is made to Figs. 21B – 21D which illustrate the relationship between features and their analytical representation as CELs and vectorized line components. Fig. 21 B shows first and second features 750 against a substrate background 760. Fig. 21C shows a map comprising a collection of pixels 770, in which CELs 780 are seen in each of the Pixels 770. CELs represent a sub-pixel crossing of edges between feature 750 and substrate 760. Features are shown as a "black" region because it typically has a relatively high coefficient of reflectivity and exhibits negative DoG. Substrate which is shown as a "white" region because it typically has a relatively low coefficient of reflectivity and exhibits a positive DoG.

Reference is now made to Fig. 21D which shows a representation of the edges of features 750 (Fig. 21B) in which the edges of features 750 are represented as edge vectors 790. In accordance with a preferred embodiment of the present invention, during the learn reference scenario described above, the location and orientation of CELs in individual pixels, for example CELs 780, are calculated for an image of an object under inspection 4 (Fig. 1A), and recorded. Collections of CELs that are substantially in the same direction are combined into connected components, and converted into edge vectors 790, and which are stored for use as a reference during the inspection scenario described hereinabove.

A preferred method for generating connected components from CELs is described in greater detail in Appendix J. A preferred method for generating edge vectors from connected components is described in greater detail in Appendix K.

Color CEL Representation

Color CELs are extensions of binary CELs. Color CELs differ from binary CELs in that they identify and define borders between two different regions in an image in which each region is one of a number of known regions having a substantially homogeneous color. Color CELs indicate the location of the border to a sub-pixel accuracy and in addition indicate which of the regions lies on either side of the CEL. The data structure of color CELs is generally understood in view of the foregoing description of binary CELs.

Each color CEL typically comprises the following fields:

x (12 bits) is the X coordinate of a CEL. Note that inasmuch as each line in an image are grouped into a data array, the y coordinate of a CEL is specified from the line in a CEL data array in which a CEL is contained.

Direction (1 bit) and edge code (3 bits) define the orientation and direction of a CEL.

last or "L" –(2 bits) defines the location of the last endpoint of a CEL along an edge of a pixel. An endpoint is the point along the edge of a pixel where intersected by a CEL.

first or "F" –(2 bits) defines the location of the first endpoint of a CEL along an edge of the pixel.

Material edge – (4 bits) defines which homogeneous color or material population exists on each side of the CEL.

It is noted that in the preferred 24 bit data structure, resolution is sacrificed (as compared to a binary CEL) in favor of identifying color populations. This is however a design choice and higher resolution may be obtained by employing a data structure having a larger number of bits. Additionally, a different meaning is assigned to edge code = 7. In this case, this edge code 7 denotes a color junction. A color junction is a pixel separating between two color CELs having material edges with different values.

Morphological Feature Representation

Each morphological feature is preferably represented by the following fields:

x (12 bits) is the X coordinate of a morphological feature. Note that inasmuch as each line in an image are grouped into a data array, the y coordinate of a morphological feature is specified from the line in a feature type data array in which a feature is contained.

data –(20 bits).typically represents one of the following types of morphological features:

OPEN END

ISLAND

3 WAY JUNCTION

4 WAY JUNCTION

5 WAY JUNCTION

BLOB JUNCTION

PIN HOLE

AN "IN" CORNER

AN "OUT" CORNER

Hardware Defect Representation

Each defect is preferably represented by the following fields:

x (12 bits) is the X coordinate of a defect feature. Note that inasmuch as elements in each line in an image are grouped into a data array, the Y coordinate of a defect is specified from the line in a defect type data array in which a defect is specified.

data –(20 bits).typically represents one of the following types of defects:

Small CEL defects

STRONG NICK

STRONG PROT

GENERAL SMALL CEL DEFECT

CEL ON SKELETON – which is a CEL located on a morphological skeleton, which is a defect typically resulting from a line being too thin

Match defects

EXCESS FEATURE – is a feature in the image of an object which is detected in an online inspection scenario (for example "inspect") which does not have a matching feature in the reference;

ILLEGAL MATCH – is a feature detected in an online inspection mode that matches to a feature in a reference, but for which the reference feature is different. For example, the online feature is JUNCTION but the reference feature is an OPEN END;

MISSING – is a feature in the reference image which does not have a matching feature in the image of an object obtained in an online inspection scenario.

Color Defect Representation Each color defect is preferably represented by the following fields:

x (12 bits) is the X coordinate of a color defect. Note that inasmuch as elements in each line in an image are grouped into a data array, the Y coordinate of a color defect is specified from the line in a color defect type data array in which a color defect is specified.

Color defect data (20) bits. In preferred embodiments of the invention color defects are filtered by watchdog tasks, for example watchdog 620 in Fig. 14, and passed directly to a defect report.

Width Defect Report

A width defect report outputs from a line width inspection process. The report is in the form of a data source which contains line width defects detected by the process. Collections of width defects are provided in a width defect data structure array.

A single width defect data structure contains the following fields:

x (12 bits) is the X coordinate of a width defect. Note that inasmuch as elements in each line in an image are grouped into a data array, the Y coordinate of a width defect is specified from the line in a width defect type data array in which a width defect is specified.

type (1 bit) specifies the type of width measurement as being one of either

SPACE SKELETON, which is a space measurement; or.

LINE SKELELETON, which is a line width measurement.

direction –(4 bits) specifies the direction of the measurement preferably from among one of the following possible directions:

Unknown direction,

East direction,

South direction,

South-east direction

South-west direction

width (15 bits) specifies the the width measurement of the width defect.

Other Principal Forms of Data in SIP

The following section describes various other principal forms of data that are used in the SIP.

Snap Image Representation

Each snap image is preferably represented by the following fields:

x (8 bits) is the X coordinate of a snap image. Note that inasmuch as elements in each line in an image are grouped into a data array, the Y coordinate of a snap image is specified from the line in a snap data array in which a snap image is specified.

r –(8 bits) which is the intensity of the Red RGB band.

g –(8 bits) which is the intensity of the Green RGB band.

b –(8bits) which is the intensity of the Blue RGB band.

A preferred method for computing using a snap data structure is described in Appendix B.

Line Envelopes

A line envelope is an envelope structure that surrounds a line, such as an edge contour, and is generally employed in the CEL to Vector comparison function, described hereinbelow with reference to Figs. 30A – 32B.

Reference is now made to Fig. 24 which illustrates a line envelope construct 800 which is termed herein an "envelope of an oriented line". An envelope of a line is a rectangle, constructed around that line. Line envelope 800 has the same

orientation as that of line 810. The perpendicular distance from the line 810 to both sides of the rectangle is denoted by a parameter, denoted width 820, while the distance from each of the endpoints of line 810 to the ends of the envelope is denoted by the parameter tang width 830. Another parameter used in the definition of an envelope is the angle tolerance parameter, which is a parameter in a CEL to Vector comparison function that ensures that the direction of a CEL is close to the direction of a line.

Reference is now made to Fig. 25, which shows line envelopes for a polyline 850. A polyline 850 is simply a collection of lines 810. As seen in the figure, a polyline envelope 860 is merely a collection of all the envelopes for each line segment 810 that makes up polyline 850.

PIM

The PIM (Point In Map) is a collection of data structures and processes that is useful for solving point in map queries. A point in map query is a query that identifies where a particular point in an image is located, and is useful to ensure that a task or function is performed on or at an appropriate point or region in the image. A description of a filter tolerance setting PIM constructed and operative in accordance with a preferred embodiment of the present invention is provided in Appendix C.

The PIM is preferably used by SIP filtering and watchdog tasks

MIM

Is a data structure that is preferably based on the PIM. It adds an additional functionality of attaching different data items to different regions. Those data items may represent, for example a threshold for computation. Use of MIM enable different events, for example features and defects, to be treated differently, as a function of location in addition to event type.

For example, a computational process that measures the area of a cluster of surface defects. If the defect is inside some predefined critical area, then a small value of the area may suffice to declare the location as being defective. However, if the defect is located outside the predefined critical area, then the same value may be passed through the MIM filter so that it is not declared as being defective. By way of additional example, assume that a nick is indicated as an additional inspection trigger.

A nick is characterized by having a height and or a width that exceed a predetermined threshold. In SIP inspection, the height and width thresholds may be dynamically adjusted using MIM, as a function of the location of the nick additional inspection trigger in the pattern of an object under inspection. It is readily appreciated that the thresholds may be input by a user through a general user interface.

Top Down Reference

A top down reference is used to define a Top Down (TD) trigger. A top down trigger holds image and type information for a region or interest around a point, as is learned during the learn scenario, and directs the SIP to inspect the region of interest in a predetermined way using an inspection function that depends on the type of top down trigger. For example, a top down trigger for a bonding pad will direct the SIP to perform a set of inspection functions on a spatial region of interest defined in the top down report which surrounds the bonding pad.

A top down report preferably comprises the following fields:

X sub -pixel (4 bits): defines a sub pixel shift of a top down report in the X direction, within the pixel for which a top down report is defined.

Y sub -pixel (4 bits): defines a sub pixel shift of a top down report in the Y direction, within the pixel for which a top down report is defined.

Index (24 bits): Defines an index of a top down report within an array of all top down windows for an object to be inspected.

Reference is made to Fig. 27 which is a simplified pictorial illustration of contours of selected elements that may exist in a typical top down window, and the shapes that they represent: a line segment 880, a circular "ball" 882, indicating a center point 884, and a rectangular pad 886, indicating open ends 888. Each of the shapes have been derived from CELs as described with reference to Figs. 21A – 21D.

Once the raw data is packed into a window, it is ready for processing. Preferably, processing functions, as described hereinbelow, operate on a window that data located in a window.

Sip Defect Report

A Sip Defect Report determines the structure of a single SIP defect. A SIP defect report contains:

A coordinate (X,Y) of the location of the defect.

A descriptive string.

Enumerated type of the defect

Unified Event

Unified events are events that are derived from multiple cameras, and that are packed into unified windows.

Each unified event preferably identifies an (X,Y) location of the pixel on which the report is defined and specifies an event type chosen from among one of the following event types:

Top Down Report

Feature

Defect

Width defect

The data associated with each unified event depends on the type of the event, and comprises the data as described above with respect to each event.

Affine Transformation

The Affine Transformation is a data source that contains information about the affine transformation used to register cameras between themselves and in the inspect mode to register between the online image of the object under inspection and its reference image. The affine 2D transformation is described above.

Window Queues

Window queues are queues of windows that are ordered and waiting for the performance of some computation by a task. Window queues the principal data source used for moving windows between tasks. A window may be pushed into the queue by one task, and after a task finishes working on a window and then it placed in another queue until popped by another task that wishes to work with that window.

Registration and Multiple Camera Considerations

In accordance with a preferred embodiment of the invention, scanner 2 in the embodiment of Fig. 1A, and 34 in the embodiment of Fig. 1B, employ multiple cameras to acquire images of an objects under inspection 4 (Fig. 1A) or panel 36 (Fig. 1B). Use of multiple cameras necessitates transforming the input of each camera by a corresponding alignment transformation. The alignment transformation is an affine transformation computed so that every point in the panel that is observed by more than one camera is mapped by the alignment transformation to the same point in the aligned coordinate system. Preferably, the aligned coordinate system is set so that a unit in this coordinate system equals about 15 microns. A point in this coordinate system is denoted by (Xa,Ya)

Scanning with many cameras obviously has consequences when designing processes that find defects. For example, a preliminary step in the inspection of a large object that is not viewed entirely by any single camera is to operate on pieces of the object that are entirely viewed in each different cameras. Then, unresolved pieces are unified and inspected. Finally, each piece from the coordinate system of a camera in which a task is performed is transformed to a unified aligned coordinate system, and the partial images are stitched together into one large object represented in an aligned coordinate system.

In accordance with a preferred embodiment of the invention, a camera model is provided that specifies the geometry of the optical head: including, for example, the number of cameras, the width of each camera (preferably expressed as a number of diodes or pixels), the overlap between adjacent cameras and an alignment transformation needed to transform a point in a given camera into a unified, camera independent coordinate system. The unified coordinate system is called the aligned coordinate system.

A description of a SIP camera model, constructed and operative in accordance with a preferred embodiment of the present invention is provided in Appendix D.

Preferably, two camera models are stored:

CAMERA MODEL, which is the camera model alignment information for a particular inspection machine at the time of inspection; and

CAMERA MODEL REFERENCE, which is the camera model of an inspection system at time of performing a learn scenario. The camera model reference may or may not be the same as the camera model at the time of inspection. Use of the double reference enables, for example, to perform learn and inspection scenarios on separate inspection systems.

In the preferred inspection scenario of the present invention, when a unified image of an object being inspected is compared to a reference, an additional registration transformation is used to register the reference and online [camera/aligned] images respectively. A preferred method for registering between images is described in U.S. Patent 5,495,535 to Harel et. al., incorporated herein by reference, which is generally adapted to identify clusters of features and to perform dynamic registration among clusters rather than by registration between individual features.

Reference is now made to Fig. 26 which is a simplified pictorial illustration of a collection of like frames 870, for example as may be collected together on a panel 36 (Fig. 1B), for inspection in accordance with a preferred embodiment of the present invention. Each of the frames is identical one to the other. Parameters and attributes preferably are learned for a single frame 870 only, and global registration coordinates are changed so that each frame may be inspected and compared to a reference individually.

Tasks

The following section discusses principal tasks that are performed by the SIP. Tasks are represented as functional blocks in the execution graphs of Figs. 19 and 20.

Input Output Tasks

Input output tasks connect the SIP process to the outside world. A more full description of input output tasks may be found in Appendix A.

Input Channel Task

An input channel task is connected to a FILE or TCP/IP channel. The input channel taskreads incoming raw data and converts the raw data into data sources.

In the preferred embodiment of Fig. 1B, the input channel task is associated with data converter and splitters 80 and 120.

Output Channel Task

An output channel task writes data sources into a designated channel, either FILE or TCP/IP. For example, in the execution graph of Fig. 2 an output channel task is associated with defect report generator 230.

Composite Task

A composite task is a task that manages a collection of tasks and data sources. When a composite task is executed it decides the order in which tasks will be executed. The selected task is the task with the highest priority among any of the tasks that are available to work when a call is made to perform the task A more full description of the runtime behavior of composite tasks is presented in Appendix A.

Measure Line Widths

Reference is made to Figs. 28A - 28B which are illustrative of a preferred method for measuring line widths. The line width of a feature, such as a bonding pad, is preferably measured with reference to a morphological skeleton 900. A line is defined by two lines 920 and 910 each of which is formed of a collection of CELs that are located on either side of skeleton 900. Line width is defined as the length of a line that is perpendicular to skeleton 900 and which touches two opposite lines 910 and 920. Fig. 28A shows line width measurement for a skeleton having substantially parallel edges. Fig. 28B shows width measurement for a skeleton having non-parallel edges.

A preferred line width method is described in Appendix E.

Line width measurement tasks are performed in each of the design rule, learn and inspect scenarios described above.

Preferably, line width measurement outputs statistics of line widths and spaces between lines for all locations in an object being inspected, and issues and error report in the event that a minimum line width is violated. Typically, line widths are associated to a line and a location, while minimum space violations are applied globally.

Learn Line Widths

A line width reference comprises a set of rectangles, each of which encloses an area in an image within which all line width measurements have approximately of the same value. In areas where the widths of the lines have a high degree of change, it is typically not possible to create a suitable line width reference. The learn line width task preferably is operative to identify the regions where a high level of change in line widths makes it impractical to form a line width reference. These areas are enclosed in rectangular areas which are identified as areas for which line widths can can not be learned.

Inspect Line Widths

Inspect line widths measures the width of lines and the spaces between lines in the image of an object under inspection. Computed widths are compared to the width of lines and spaces between lines for the corresponding position in a reference image. A registration transformation is used to correlate between the respective positions in inspected and reference images. Any mismatch between a computed line width in an online image of an object, and a corresponding acceptable range for line widths as designated in the line width reference for the given location, is reported as a line width defect.

Registration

The registration task produces a transformation between an online coordinate system for images of objects under inspection and a reference coordinate system. Preferably, if a number of cameras are used, a transform is provided for each camera, although it is appreciated that for ease computation only one camera, for example the central camera, may be used to perform registration operations.

Reference is made to Fig. 29 which is a pictorial illustration of an example of a transformation from an online coordinate system to a reference coordinate systems, and the inverse from reference coordinate system to an online coordinate systems. The left-hand coordinate system 950 is an on-line coordinate system and the right hand coordinate system 960 is a reference coordinate system. The illustrated

example shows a rigid transformation comprising only rotation and shift. It is appreciated, however, that the registration transformation need not be rigid and that it may generally comprise any suitable transformation.

A preferred registration method is also described in Appendix E.

Learn Registration

The learn registration task creates a reference of features that serves as reference for registration. The (X,Y) coordinates for each morphological feature, and its respective feature type are preferably stored. Certain types of features are defined as "stable" meaning that their location in an object to be inspected preferably is constant to within a very small margin, and registration is preferably performed using stable features.

Inspect Registration

The inspect registration receives as an input an input array data structure, and outputs a transform data structure for an affine transformation. In addition, data structures are output for excess defects, missing defects, and illegal matches.

Objects being inspected are preferably at least partly inspected using a compare to reference approach in which during the learn scenario, prior to inspection, a reference is created for a believed-to-be "perfect" object. The perfect object is preferably obtained either by scanning a golden board chosen to be particularly free of defects or by learning from CAM data. During an inspection scenario, an online image from a scanned object is compared against the perfect reference, and a decision is typically made as to whether errors on the online panel cause is to be too different to the reference panel.

Typically it is not possible to do a straightforward comparison between images of the reference and online panels since the location of the online panel on the scanner may differ slightly from the location at which the reference image was generated. Thus the coordinate of a given feature on the golden panel is typically different from the coordinate of the same feature on the online panel.

The registration task uses features to compute a registration transformation that maps points from the on-line coordinate system to the reference coordinate system.

A brief explanation of a preferred method used in the inspection of BGAs follows. First the input features are unified into clusters of generalized features, which tries to overcome an inherent instability of morphological features on BGAs. The same feature (say, a circle) for example may result in one island feature in one scan and in two, very close together, open end features in another scan. Two open ends can be defined as a single generalized feature, and a matching rule can be defined that specifies that it is legal to match this generalized feature with an island feature. The set of generalized features and the match rules are typically specified in a special configuration file called a matcher table. It is also possible to define in configuration files the unifying distance for unification of different morphological features into one generalized feature.

The registration module also is preferably operative to identify reference features that are not found in the set of on-line features as being "missing", and on-line features that not found in the set of reference features as "excess". The registration task preferably also identifies illegal matches. An illegal match occurs when a feature is detected in the online coordinate system that is matched to a reference feature (i.e., after transforming the reference feature using the registration transformation, the feature is found to be close to an online feature that has a very different type). A typical example of illegal match is a match between Junction and island.

Excess/missing/and illegal match features are reported as excess/missing defects.

Split Registration Transformation

In one preferred embodiment of the present invention, the registration task works only on camera features that are located in the field of view of the central camera. The split transformation task takes the registration transformation and splits it into three registration transformations, one for each camera. In another embodiment, in which registration data is obtained for each of the cameras, this task is not be needed and the registration task is capable of working on aligned features.

Snap Data Packing

In the snap data packing task, input image data is processed and windows are constructed which comprise full image data for a desired region. The windows are computed by clustering together input color defects so that each snap window will contain snap data around a specific cluster of color defects. After snap data is packaged into a window that preferably also comprises color defect data, the window is pushed onto an output queue. A preferred method for packing snap data is described in greater detail in Appendix F.

Building PPM File For Viewing Snap Data

In a preferred embodiment of the present invention, a snap image is constructed for selected defects. Input data is processed and a set of output files containing RGB images of contiguous snap areas are produced. A preferred description of this process is provided in Appendix G.

Watchdogs

Watchdogs compare data inside input data sources against a specified PIM. If certain types of predetermined data are found to exist inside prohibited areas as described in the PIM, then the offending data is converted into SIP defect. SIP defects are accumulated in SIP defects data structure which are packaged into a window, and the window, with all its defects, is pushed to an output queue of windows. One or more of the following features and defects are preferably watched by watchdogs: color defects, hardware defects, line width defects, morphological features.

Task Packer

The packer packs features, CELs and defects into specified windows (rectangular areas) that form a region of interest for additional downstream processing. Windows are preferably specified for each top down event as well as each width defect, excess/missing defect, and hardware defect.

Top down windows that form part of the top down reference are preferably constructed during the learn scenario, and are preferably transformed into the

on-line coordinate system (using registration transformation) before CELs and features are packed inside of the top down windows.

In order to minimize the number of windows created and yet enclose ALL the interesting information (top down windows and windows around defected areas) the following scheme is typically used:

First pack data for all top down windows add each packed window to the queue of packed windows.

Consider all defect and line width defect reports:

1. Take the first defect not inside a previously created window (ordering is done in increased y and then increased x coordinates).
2. Create a window centered at the chosen defect, pack all data inside this window and add it to the queue of packed windows.
3. If there are more defects not within previously created windows - go to step 1.

At the end of the packing process all data (CELs, Color CELs, Defects, Features, Width defects) is packed in one or more windows, and the windows are output to a window queue. It is noted that windows may overlap, in which event data items within the overlap region are packed onto each of the overlapping window.

Raw Data Reference Building

This task takes the input raw data and creates a reference of the raw data that is suitable for fast access during the inspect scenario. Further details of preferred structures and methods associated with the windowing of reference raw data are provided in Appendices H and I.

Filter

This task is attached to a PIM for filtering. We assume that the PIM is defined in reference aligned coordinates. The task receives input data for color CELs, features, width defects, defects and affine transfers, and considers each element of the input data sources. First the element is transformed into reference (aligned) coordinate system using the input Affine transformation. If no input transformation is specified,

the alignment transformation is used to transform element from camera to aligned coordinate systems.

Next each element from the input data is considered. This element is transformed into the reference aligned coordinate system and checked if the element is inside designated filter zones of the PIM. A configuration file determines whether to filter this element or not. If the element is not filtered, it is pushed into output data source. The value of the configuration variable typically controls whether to filter if an element is inside a designated zone, or to filter if an element is outside a designated zone.

Extract Info From Windows

This task is operative to pop windows out of an input data queue, and to extract data from that window as needed. Upon completion, then the window is pushed into the output queue.

Produce Reference of Zones and of Stable Features

This task writes auxiliary data from an input window with respect to data that typically comprises: (a) PIM regions of various types and (b) All features declared as "stable" features by previous functions that operated on a window.

Balls Measurements

This task writes auxiliary data from an input window. The data typically comprises a file containing information on all circles found in windows and the two others contains statistics on distribution of circles in the panel. Further details of a preferred method for performing this task is provided in Appendix J.

Test Manager

This task preferably pops a window from an input window queue and runs a function associated with that window using fundamental image information data included in the window. Based on the output of the function it decides what to do as follows:

TABLE 1. Action of test manager for each return status from function.

return code	action
ERROR	Window deleted and task return with an error.
OK	Window is deleted. Task continue to pop next window
FORWARD	Window is sent to forward queue.

FORWARD SUB WINDOWS After completion of the task each of the sub-windows with a main window is sent to a forward queue, while the original input window is preferably deleted. In a preferred embodiment of the present invention, a sub-window may be formed, for example around individual bonding pads in a bonding area window.

Preferably, if no function is attached to a window, then the return status is OK.

Multi Slice Manager

The multi slice manager is preferably operative to receives windows including unresolved data from different cameras. Preferably, the windows are located in an aligned coordinate system. The multi slice manager unifies all windows having the same window identification number and stores the unified window in memory. A unified window that encloses all sub windows is created and then a function attached to the unified window is executed. Based on the output of the inspection function, it decides what to do as follows:

TABLE 2. Action of multi slicer manager for each return status from function.

return code	action
ERROR	Window deleted and task return
with	an error.
OK	Window is deleted. Task continue
to	pop next

	window
FORWARD	Window is send to forward queue.
FORWARD SUB WINDOWS	As in ERROR
none of the above	As in ERROR.

It is appreciated that, typically, if no function is attached to the window, the return status is OK.

Defect Writer

The defect writer task preferably accepts four file names from configuration files:

(a) Output window file name: which is a file containing windows. A value of "none" indicates that the file name is not a valid file name and that this file is typically not produced.

(b) Defect window file name: which is a file containing defective windows. A value of "none" indicates that this is not a valid file name and that this file is typically not produced.

(c) Defects file reference coordinates: which is a file containing defects in reference aligned coordinates. A value of "none" indicates that this is not a valid file name and that this file is typically not produced.

(d) Defects file online coordinates: which is a file containing defects in online aligned coordinates. A value of "none" indicates that this is not a valid file name and that this file is typically not produced.

The Defect Writer pops windows out of an input queue and preferably proceed as follows

1. If the window contains defects:

If it is desired to produce file with defective windows or an output file then the window is stored for later processing. Otherwise, the window is stripped from all its data except defects and is stored for later processing.

2. If the window does not contain defects:

If it is desired to produce an output window file then the window is stored for later processing, otherwise the window is deleted.

At the end of a scan all windows previously stored are processed as follows:

First, all defects from all windows are accumulated. In the accumulation step, two vectors of defects are created:

1. A first vector containing defects in reference aligned coordinates.

The defects inside each window are transformed into reference coordinate system by using a Self to reference affine transformation which preferably exists for each window and transfers content of the window to a reference aligned coordinate system. Defects are added to the vector only if they do not fall inside the mask area defined by the PIM specified in an input PIM data source. If no PIM is defined, all defects are added to the vector.

2. Vector containing the same defects in online aligned coordinates.

This vector is produced from the first vector by applying the registration transformation specified as input to the task. Note that the identity transformation is used when no input affine transformation is defined.

If a valid output file containing reference coordinates is defined then the first vector is written to that file. If a valid output file containing online coordinates is defined then the second vector is written to that file.

Each of the defect files contains a set of lines, one line for each defect. Each line typically has the following format:

X Y - defect type - frame index

where X,Y are the coordinate of the defect (either in reference aligned or in online aligned coordinate systems);

defect type is the enumeration of the defect; and

and frame index is the index of the frame containing the defect.

Next, if there is a valid defects window file for the named file, then we write all defective windows into that file. This is particularly useful to visualize defective windows.

Finally, in the case of a valid output window file for the name file, the function attached to the window is first run, after which a decision on how to proceed is made by considering the return status of the function:

TABLE 3. Action for each return status from function

return	code action
ERROR	Task return with an error. All windows are deleted.
OK	Window is written into the used file.
FORWARD	As in the OK case.
FORWARD SUB WINDOWS	All sub windows within the window are written to the used file.
none of the above	As in ERROR.

It is appreciated that if no function is attached to the window, the return status is typically OK.

Draw Aligned CELs

The draw aligned CELs task considers each input CEL. If the CEL is not inside a clipped region that is specified in a configuration file, then it is converted from camera to aligned coordinates and written into a file in a special graphic format that makes the file suitable for viewing.

This task is useful for checking alignment transformation within an inspection system. Preferably, the task is run for all cameras and output files are displayed one over the other. A good alignment transformation is a transformation that transforms CELs from two overlapping cameras so that the CELs from the first camera are located exactly over CELs from the second camera in regions where the two cameras overlap.

Functions

A function is an encapsulation of an inspection method for use on image information contained in an inspection window. Functions are activated by a window oriented task, preferably either a test manager task or multi slice manager task as explained above, preferably in a predetermined order in order to inspect the image data contents of a window. The activating task applies a function to data in the window and executes the function with the specified window as an argument. The function performs its computational task on the window and returns one of three return codes described below. Based on the return code the calling task decides what to do next.

Each function preferably receives all of the parameters needed for computation from a configuration file, and extracts from the window whatever image data information that it needs for computation. Preferably, functions are modular and chained together when more complex computations on a given window are necessary.

A function preferably returns one of three possible values:

OK, which indicates that the function succeeded in its computation.

ERROR, which indicates that the function failed in computation (for example because of a bug or other serious problem); or

FORWARD, which indicates that the function succeeded in its computation but indicates that additional computation is needed in order to complete the computational task. Completion of the task is performed by forwarding the window to a forward destination and assigning to the window an additional function to the window. The activating task typically knows how to send the window to its forward destination and how to activate the forward function.

FORWARD SUB WIDOWS, which indicates that the function succeeded in its computation but indicates that additional computation is needed in order to complete the computational task. Completion of the task is performed by forwarding all sub windows in the window to a forward destination and assigning to each sub-window an additional function to that sub-window. The activating task typically knows how to send the sub-windows to their forward destination and how to activate the forward functions.

No Operation (NOP) Function

This function does nothing, and preferably returns an OK value.

Composite Function

This function preferably runs a list of functions provided in a configuration file. The functions are executed in the order specified in a configuration file.

Forwarding Function

This function obtains the name of a forward function and a forward destination for additional processing from a configuration file. The forwarding function preferably accepts flags that specify special exceptions to forwarding, typically including:

Forward Only if Inside Camera exception, which is operative to ensure that a window is forwarded only if it is completely within camera boundaries.

Forward Only if Defects Found exception, which is operative to ensure that a window is forwarded only if it contains defects.

Forward Sub Windows exception, which has a true/false flag and is operative to return status such that if the value of the flag is true, then a forward sub-window status is returned. Otherwise a forward status is returned.

Windows are preferably not forwarded if the following conditions are met:

1. There are no defects in the window, provided however that the window is flagged to be forwarded only if defects are present in the window;
2. The window is not located fully inside camera boundaries, provided however that the window is flagged to be forwarded only if the window is fully inside a camera boundary.

Transform Window To Reference Coordinates Function

The transform window to reference coordinates function removes non-transformable data items from window and transforms the coordinates of a window from online image coordinates to a reference coordinate system.

Remove Non Transformable Data From Window Function

The remove non transformable data from window function preferably removes non transformable image data, for example CELs, that exist in a window.

Watchdog Function

The watchdog function scans all input CELs within a window. Every CEL that is found inside a quadrilateral enclosing a reference window is reported as a defect. This function is typically used for regions in which it is desired to ensure the

absence of any features. For example, in the context of BGA inspection, a watchdog function may be applied to region defined in a top down window as having to be absolutely clear of any features.

Defect Filtering Inside a Window Function

The defect filtering inside a window function is attached to a PIM data source. The function receives a PIM referenced to an aligned coordinate. When executed, the function transforms input defects to reference aligned coordinate system, and checks the aligned defect against a PIM. If the defect is found inside a mask zone (or zones) specified by a PIM, then the defect is filtered out, otherwise it is pushed into an output list of defects.

Connected Components Function

The connected components function creates connected components from CELs, (including both binary CELs and color CELs) that exist inside the window. The resulting connected components are placed in placed in a window, preferably for further processing by the Vectorizer Function, the description of which follows. A preferred method for performing this task is described in Appendix K.

Vectorizer Function

The vectorizer function takes connected components and approximates each connected component by straight lines so that the deviation of the connected component and the straight line approximation is no more than a prescribed tolerance that is specified from a configuration file. A full description of a preferred method for performing this task is provided in Appendix L.

Ball (Circle) Modeling and Inspecting Functions

Ball modeling functions as are preferably used in the context of BGA inspection are generally described below and in Appendix J. Ball modeling is exemplary of a type of modeling that is suitable for use in inspecting online images of objects under inspection in the context of top-down windows.

Ball Modeling with Connected Components

Ball modeling processes connected components that are found inside an inspection window. The function is operative in the learn stage to prepare a reference model, and in the inspect stage to determine the characteristics of balls in an online image of an object under inspection. If connected components do not already exist, then the function produces connected components from CELs. An inspection window is supposed to contain only balls. Accordingly, if a connected component in a window is closed it is assumed to be a single ball, however if the connected component is open, it is assumed to be a part of a ball. The function analyses the connected component to see if it is indeed a ball, for example by testing for a center point and then measuring a radius along the contour of the connected component to determine if is within a predetermined tolerance. Information on the circularity of a single connected component is held in a data structure denoted a generalized circle.

Prepare Ball Window For Reference

Prepare Ball Window for Reference removes from the window all objects other than the balls.

Compare Reference Balls To Online Balls

The Compare Reference Balls to Online Balls functions compares balls in a reference image to balls in an online image for an object being inspected. Preferably, the location and radius of online balls is compared to reference information for the balls. Balls in an online image are deemed to match balls in the reference image, if the online ball has a similar radius, with a prescribed tolerance, and is found to be within a prescribed distance from the reference. Any excess balls, and any missing circles are recorded as defects.

CEL to Vector Comparison

The CEL to Vector comparison function compares reference vectors (polylines) obtained from a reference window to CELs existing in an online image of an object 4 (Fig. 1A) under inspection, and based on the comparison determines whether a putative defect is a real defect or a false alarm. CEL to Vector comparison is

particularly suited for the inspection of hardware defects, excess / missing defects, illegal matches defects, and morphological features. Additionally, it is used in the inspection of certain top-down windows, for example bonding pads.

Reference is now made to Fig. 30A which is a pictorial illustration of an inspection window 1000 in which reference vectors, or polylines 1010, defining a reference feature 1020, are seen. Polylines 1010 are each associated with envelopes 1030.

Reference is additionally made to Figs. 30B which is a pictorial illustration of an inspection window 1002 in which a collection of online CELs 1040 calculated from an online image of an inspected object 4 (Fig. 1A) are superimposed over the polylines 1010 of Fig. 30A. It is readily appreciated that the collection of CELs 1040 is registered to be in alignment with reference polylines 1010. The collection of CELs 1040 is identified as a candidate defect, for example by defect identifier 13, because of a putative protrusion 1050, and the CEL to Vector function is applied in order to verify whether the putative protrusion is a candidate defect or a real defect.

As is seen in the illustration, putative protrusion 1050 extends outside the region defined by envelopes 1030. Accordingly, in accordance with a preferred embodiment of the invention, the CEL to Vector outputs a defect flag for inspection window 1002.

Reference is additionally made to Fig. 30C which is a pictorial illustration of an inspection window 1004 in which a collection of online CELs 1060 is calculated from an online image of an inspected object 4 (Fig. 1A), which is different than the inspected object 4 (Fig. 1A) from which CELs 1040 (Fig. 30B) are calculated. It is readily appreciated that the collection of CELs 1060 is registered to be in alignment with reference polylines 1010. The collection of CELs 1060 is identified as a candidate defect, for example by defect identifier 13, because of a putative protrusion 1070. The CEL to Vector function is applied in order to verify if the putative protrusion is merely a candidate defect or a real defect.

As is seen in the illustration, despite putative protrusion 1070, all of the collection of CELs 1060 lies within the region defined by envelopes 1030. Accordingly, in accordance with a preferred embodiment of the invention, the CEL to

Vector function does not output a defect for CELs 1060 since they are not outside envelope 1030.

However, in Fig. 30C there is also seen a second collection of CELs 1080 which lies entirely outside envelopes 1030. Each of the CELs in second collection 1080 is flagged as a defect. It is readily appreciated that CELs located outside envelopes, such as CELs 1080, may be filtered through a defect handler function, and reported as a defect or filtered as being not of interest.

The CEL to Vector function preferably proceeds according to the following sequence:

An inspection window, for example inspection window 1000, is checked for the presence of CELs and to ensure that the number of CELs in the window does not exceed the number of CELs exceed a maximum number of CELs that is permitted to be contained in the inspection window. Preferably, the number of CELs is calculated by counting the midpoint of each CEL. It is appreciated that the parameter is variable and may be set or changed in response to the location of the window in an image.

Inspection window 1000 is checked for the presence of a reference polyline 1010. If inspection window 1000 does not include a polyline 1010, then a defect is returned.

Preferably, polyline 1010 and collections of CELs 1030, 1040 and 1060 are microregistered. Reference is now made to Figs. 31 – 32B which show and describe a preferred method of micro-registration.

Micro registration preferably proceeds as follows:

Step 1100: each CEL 1110 located in an inspection window 1120 is checked, and those CELs 1110 that are inside one or more envelopes 1130 surrounding reference ployline 1240 are identified and appropriately tagged;

Step 1200: each of the CELs 1110 that are inside an envelope 1130 are checked for direction. Each CEL whose direction, within an angular tolerance range provided by a configuration file based on inspection requirements, is the same as a polyline1240, is denoted as a "matching" CEL 1210.

Step 1300: A least squares calculation is performed on the distance between each matching CEL 1210 and its neighboring polyline, and the collection of

CELs 1210 in the online image is moved relative to polyline 1140 until the value is minimized. When the value is minimized, the images are microregistered, as seen in Fig. 32B.

Following microregistration, a CEL to Vector comparison is conducted as follows:

Envelopes 1030 (Fig. 30A) are constructed around the reference polylines with width and equal to the value of a configuration variable .

Subsequently, CELs 1040 are compared to envelopes 1030. If a CEL 1040 is not found in any envelope a defect is reported. Preferably, each envelope records the number and length of all CELs 1040 that are present inside the envelope 1030. If no CELs are recorded in a given envelope in an inspection window 1000, a defect indicating that the affected envelope is empty is reported for the midpoint of the envelope.

Create Reference Windows From a Bonding Area

A function to create reference for bonding area is preferably performed in BGA inspection. The create reference for bonding area function is operative to analyze connected components and to find connected components that are part of a functional entity called "power line". User defined configuration variables, adapted to the particular structure of power lines, are used in the decision making process. Upon the detection of power lines, calculations are made with respect to three zones of inspection interest:

- critical, which are zones that preferably exhibit strictly defined attributes such as location and material
- non critical which are zones that have a large degree of latitude with respect to necessary attributes and location
- unstable which are zones whose position is likely to change without being considered defective. Unstable zones are inspected only if requested by a configuration variable.

A function to create bonding pad reference sub-windows is also preferably operative in BGA inspection, and is suitably employed to detect whether an inspection window encloses any bonding pad. The function analyzes all connected

components in a window, and identifies those connected components that are part of bonding pad. Critical, non-critical and unstable zones are preferably identified.

In the absence of a configuration variably requiring computation of unstable zones, contours for homogeneous color populations (identified from color CELs) are morphologically dilated , in a preferred embodiment for BGA inspection, to separate solder mask regions from all other materials.

Cell to Vector Error Classification

Typically all CEL to Vector errors are considered as defects. However, sometimes a CEL to vector may indicate the presence of a "surface defect". A surface defect is an extraneous connected component which that was present in the reference data.

Reference is again made to Fig. 30 C. Assume that reference polyline 1010 is a bonding pad. In accordance with a preferred embodiment, if a collection of CELs, such as collection 1250, is detected to have a polarity that is opposite to the polarity of a polyline 1010 which corresponds to the desired bonding pad in a reference image, then a "pinhole" is indicated. Polarity is the orientation of the CELs or polyline respectively.

Post Processing

Reference is now made to Fig. 33 which is a flow chart in accordance with a preferred embodiment showing the operation incorporating image post processing system, such as may be operative as a task in image post processor 26 (Fig. 1A). The image post processor task depicted in Fig. 33 operates as a defect filter that outputs a defect in the event that small defects are found by virtue of their quantity, as opposed to their quality, necessitate issuance of a defect report. For example in the context of BGA inspection, individual pinholes in metal coatings on bonding pads may be acceptable, however if the quantity of pinholes exceeds a threshold, then the BGA is deemed defective.

In this embodiment defects in an image of an object are specified as either regular or minor defects. Regular defects in an image, corresponding to regular

defects in the object, are typically not tolerated, while a small number of minor defects, corresponding to minor defects in the object, typically are tolerated. In the example provided above, a pinhole defect may be considered a small defect, while a missing feature, such as a missing bonding pad, may be considered a regular defect.

Step 3305: A user specifies the criteria used for defining defects in an image. Screen displays similar to those of Figures 34-42 allow a user to specify defect criteria.

Step 3310: If any type of minor defect is tolerable up to some multiple, then for each type of minor defect a user specifies the maximum number (x) of each type of minor defect than can be tolerated. The number (x) may be one defect or more than one defect. In effect, an amount of x+1 of any particular type of minor is not tolerated.

Step 3315: The image is inspected, preferably in accordance using one or more the image processing methods shown and described with reference to Figs. 1A – 32B of the present invention.

Step 3320: A decision tree determines whether defects were found.

Step 3325: If no defect is found, then the image is deemed OK.

Step 3330: A further decision tree is followed to classify the defect as either a regular or minor defect.

Step 3335: If a regular defect is present, then a defect report is issued.

Step 3340: Each type of minor defect is counted.

Step 3345: If the number (n) of a minor defect type is less than the number of permitted minor defects of the given type, then no defect report issues. However if the number of the type of minor defect is greater than the tolerated number of permitted minor defects of that type, i.e. $n>x$ then a defect report is returned at step 3335.

It is readily appreciated that a user of an image processing system including a minor defects post processor as described with reference to Fig. 33 typically determines the parameters and criteria for inspection of an image, and the same or separate parameters may be employed for inspection of user specified areas of interest, and for identification of defects by the signal processor.

It is appreciated that the aforementioned description of a post processing unit is not intended to be limiting to the aforementioned method, but rather other post processing methods may be employed. Suitable post processing methods may include, for example, functions to further analyze regions of interest surrounding candidate color defects, functions operative to identify and classify defects resulting from oxidation and photoresist residue, and other suitable defect candidates that may be filtered by additional image processing of selected sections of a robust image.

General User Interface (GUI)

It is appreciated that an image processing system in accordance with the present invention is particularly flexible, and enables numerous inspection parameters to be identified and controlled, including by way of example only and not by limitation, general parameters, calibration, repeat, Ip, color and light.

Inspection criteria are also preferably specified by the user. Such parameters may comprise, by way of example and not by limitation, the type, location and contents of top-down windows for balls, targets, bonding pads and cavities, power lines and the like. Additional parameters may also be provided such as minimum line widths. In accordance with the present invention, the various parameters are preferably correlated to particular regions on an object to be inspected.

Reference is now made to Figures 34-42 which are examples of screen displays generated during user definition of criteria for inspection of an image by an image processing system constructed and operative in accordance with a preferred embodiment of the present invention.

Figure 34 is a screen display allowing a user to specify general inspection criteria.

Figure 35 is a screen display allowing a user to specify bonding inspection criteria.

Figure 36 is a screen display allowing a user to specify target inspection criteria.

Figure 37 is an example of a screen display allowing a user to specify criteria for inspection of power lines.

Figures 38-39 are examples of screen displays allowing a user to specify criteria for inspection of balls.

Figs. 40-42 are examples of screen displays allowing a user to specify advanced criteria and parameters for inspection of an image of an object.

In accordance with a preferred embodiment if the invention, a two part general user interface is provided for defining regions of interest in an object to be inspected, such as a BGA. In one part of the user interface, a graphic interface is provided in which a graphic representation of the object to be inspected is provided, and the user with the assistance of a pointing device, typically a mouse, designates on the graphic representation types of regions of spatial interest for inspection. In the second part, parameters are provided for each of the regions of spatial interest that are designated.

It is appreciated that the software components of the present invention may, if desired, be implemented in ROM (read-only memory) form. The software components may, generally, be implemented in hardware and/or DSP units, if desired, using conventional techniques.

It is appreciated that the particular embodiment described in the Appendices is intended only to provide an extremely detailed disclosure of the present invention and is not intended to be limiting.

It is appreciated that various features of the invention which are, for clarity, described in the contexts of separate embodiments may also be provided in combination in a single embodiment. Conversely, various features of the invention which are, for brevity, described in the context of a single embodiment may also be provided separately or in any suitable subcombination.

It will be appreciated by persons skilled in the art that the present invention is not limited to what has been particularly shown and described hereinabove. Rather, the scope of the present invention is defined only by the claims that follow: